

# **Лабораторная работа №11**

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

Латыпова Диана. НФИбд-02-21

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>4</b>	<b>Контрольные вопросы</b>	<b>16</b>
<b>5</b>	<b>Выводы</b>	<b>18</b>

## Список иллюстраций

3.1	Скрипт 1 задания . . . . .	9
3.2	Результат 1 скрипта . . . . .	9
3.3	Скрипт 2 задания . . . . .	11
3.4	Код на C++ . . . . .	12
3.5	Результат 2 скрипта . . . . .	12
3.6	Скрипт 3 задания . . . . .	13
3.7	Результат 3 скрипта . . . . .	14
3.8	Скрипт 4 задания . . . . .	15
3.9	Результат 4 скрипта (1) . . . . .	15
3.10	Результат 4 задания (2) . . . . .	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:

- `-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`.

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 в `o` 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  $\infty$  (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

Для начала я перешла в папку lab11:

```
cd ucheba/lab11
```

1. Запустила редактор emacs 11\_1. Здесь написала скрипт 1 задания (рис. 3.1)

**Скрипт 1 задания:**

```
#!/bin/bash
while getopts "i:o:p:cn" opt
do

case $opt in
    i)inputfile="$OPTARG";;
    o)outputfile="$OPTARG";;
    p)sample="$OPTARG";;
    c)reg="";;
    n)line="";;

esac

done
grep -n "sample"inputfile" > "$outputfile"
```



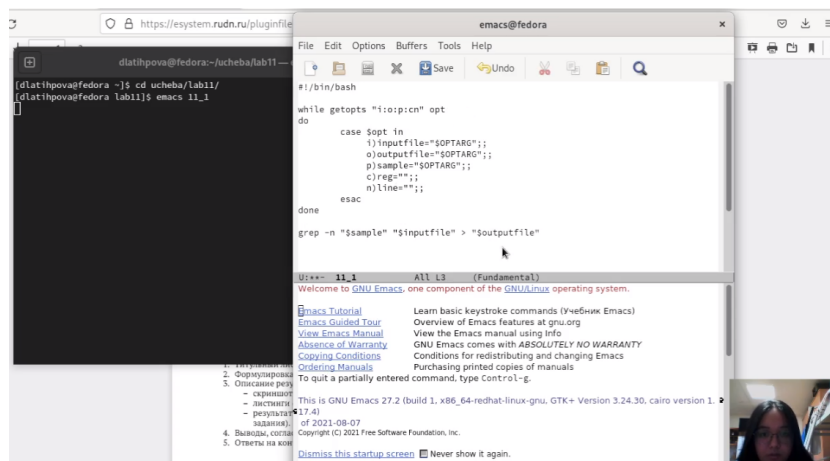


Рис. 3.1: Скрипт 1 задания

Сначала предоставила права на выполнение файла 11\_1. И запустила файл(рис. 3.2):

- 1 **chmod +x 11\_1**
- 2 **./11\_1 -i conf.txt -o result.txt -p n etconf -c -n**
- 3 **ls**
- 4 **cat result.txt**

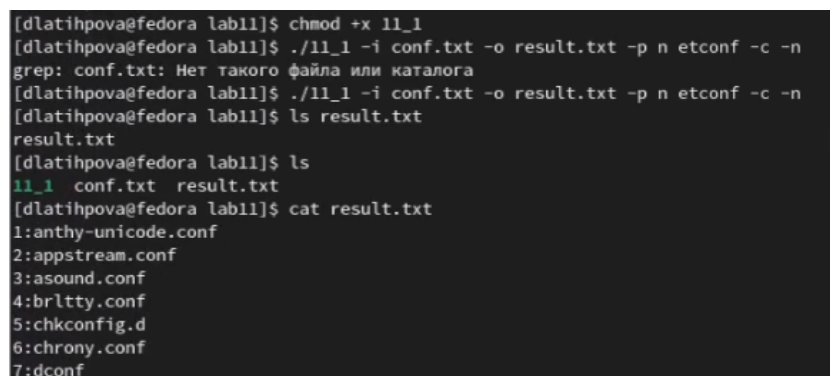


Рис. 3.2: Результат 1 скрипта

Все корректно работает.

2. Запустила редактор emacs 11\_2. Здесь написала скрипт 2 задания (рис. 3.3)

**Скрипт 2 задания:**

```

#!/bin/bash
RES=result
SRC=main.cpp
if [ "SRC" -nt "RES" ]
then

echo "Creating $RES ..."

g++ -o $RES $SRC

fi
./$RES $1
ec=$?
if [ "$ec" == "1" ]
then

echo "input > 0"

fi
if [ "$ec" == "2" ]
then

echo "input = 0"

fi
if [ "$ec" == "3" ]
then

echo "input < 0"

fi

```

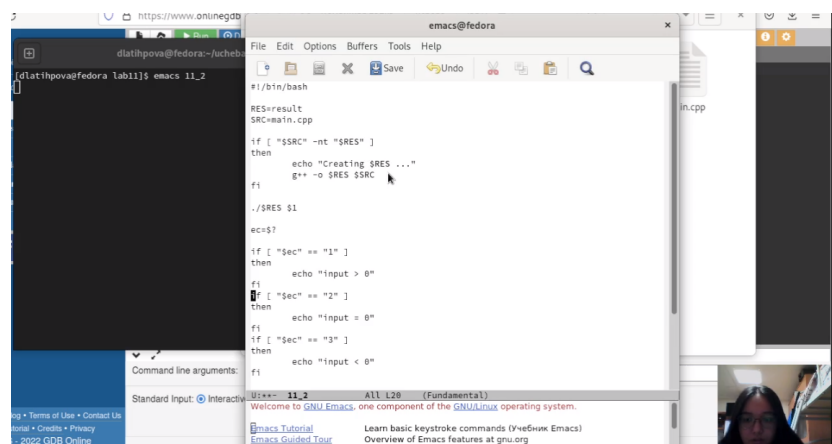


Рис. 3.3: Скрипт 2 задания

Код C++(рис. 3.4):

```

#include
using namespace std;
int main(int argc, char *argv[])
{ if (atoi(argv[1]) > 0) exit(1);

else if (atoi(argv[1]) == 0) exit(2);

else exit(3);

return 0;

}

```

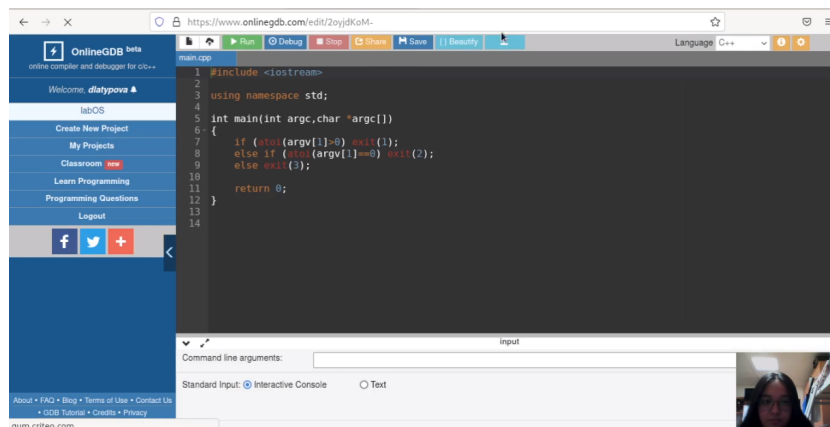


Рис. 3.4: Код на C++

Снова сначала предоставила права на выполнение файла 11\_2. И запустила файл(рис. 3.5):

- 1 **chmod +x 11\_2**
- 2 **./11\_2 1**
- 3 **./11\_2 0**
- 4 **./11\_2 -2**

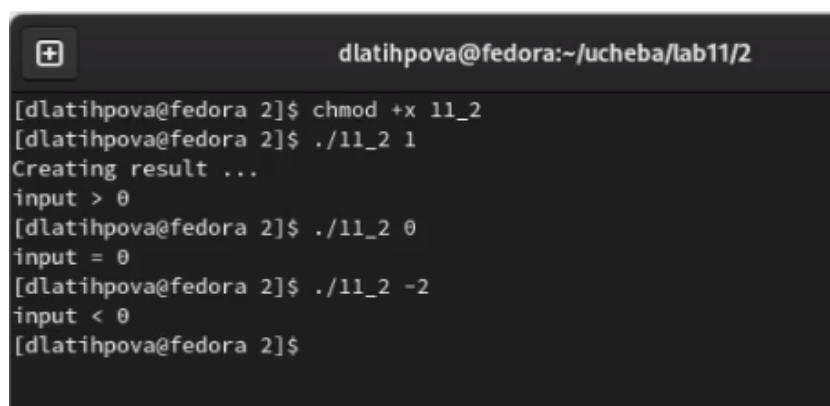


Рис. 3.5: Результат 2 скрипта

3. Приступила к написанию 3 скрипта(рис. 3.6):

**emacs 11\_3**

**Скрипт 3 задания:**

**#!/bin/bash**

```

while getopts c:r opt
do

case $opt in
    c)n="$OPTARG"; for i in $(seq 1 $n); do touch "$i.tmp"; done;;
    r)for i in $(find -name "*.tmp"); do rm $i; done;;

esac

done

```

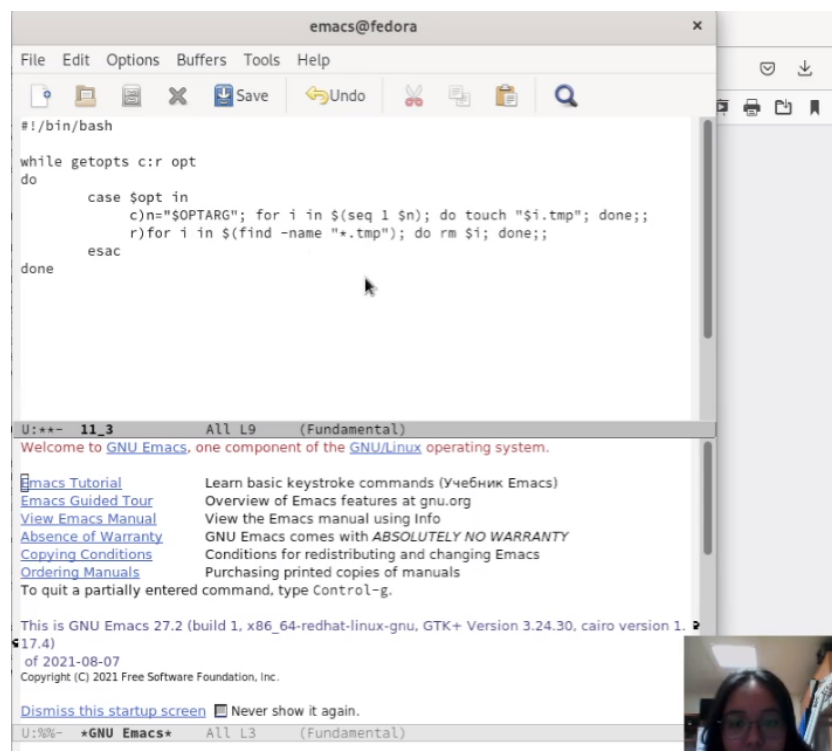


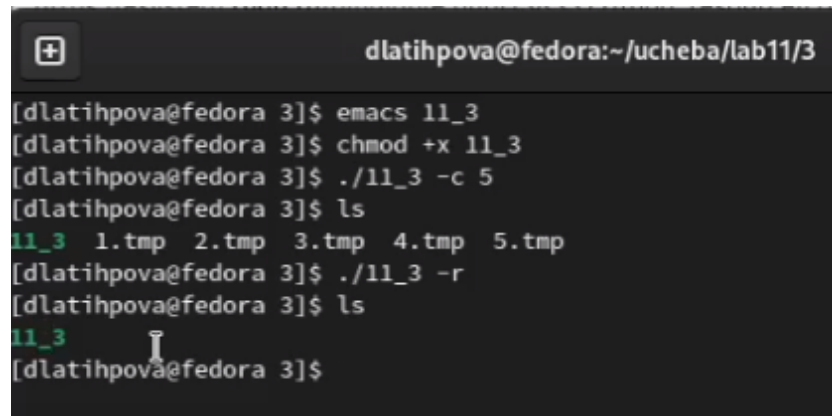
Рис. 3.6: Скрипт 3 задания

Снова предоставила права на выполнение файла 11\_3. И запустила файл(рис. 3.7):

- 1 **chmod +x 11\_3**
- 2 **./11\_3 -c 5**
- 3 **ls**

4 ./11\_3 -r

3 ls

A terminal window titled 'dlatihpova@fedora:~/ucheba/lab11/3' showing the execution of a script. The user runs 'emacs 11\_3', 'chmod +x 11\_3', and './11\_3 -c 5'. Then they run 'ls' and see '11\_3 1.tmp 2.tmp 3.tmp 4.tmp 5.tmp'. Finally, they run './11\_3 -r' and 'ls' again, and only '11\_3' is listed.

```
dlatihpova@fedora 3]$ emacs 11_3
dlatihpova@fedora 3]$ chmod +x 11_3
dlatihpova@fedora 3]$ ./11_3 -c 5
dlatihpova@fedora 3]$ ls
11_3 1.tmp 2.tmp 3.tmp 4.tmp 5.tmp
dlatihpova@fedora 3]$ ./11_3 -r
dlatihpova@fedora 3]$ ls
11_3
dlatihpova@fedora 3]$
```

Рис. 3.7: Результат 3 скрипта

Скрипт корректно работает.

4. Запустила редактор emacs 11\_4. Здесь написала скрипт 4 задания (рис. 3.8)

**Скрипт 4 задания:**

```
#!/bin/bash
while getopts :p: opt
do

case $opt in
    p)dir="$OPTARG";;
esac

done

find $dir -mtime -7 -mtime +0 -type f > archiv.txt
tar -cf result.tar -T archiv.txt
```

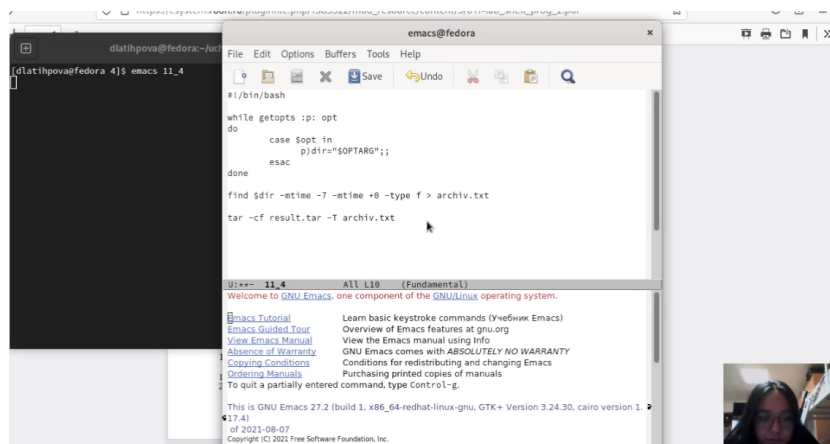


Рис. 3.8: Скрипт 4 задания

Снова предоставила права на выполнение файла 11\_3. И запустила файл(рис. 3.9)(рис. 3.10):

1 **chmod +x 11\_4**

2 **./11\_4 -p /home/dlatihpova/**

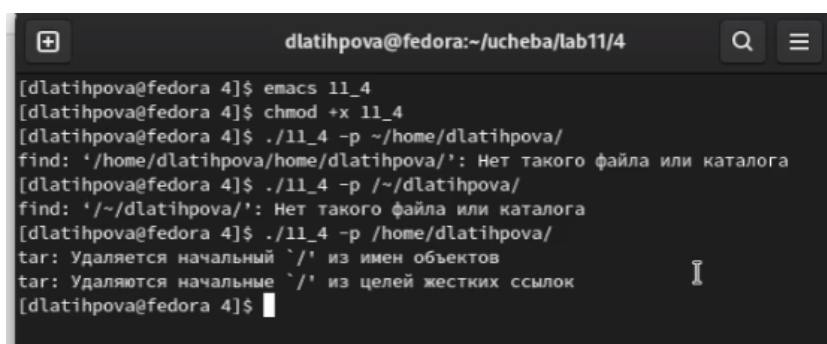


Рис. 3.9: Результат 4 скрипта (1)



Рис. 3.10: Результат 4 задания (2)

## 4 Контрольные вопросы

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

Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные 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?

При перечислении имен файлов текущего каталога можно использовать следующие символы: `*` — соответствует произвольной, в том числе и пустой строке; `?` — соответствует любому одному символу; `[c1-c1]` — соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. `echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` — выведет все файлы с последними двумя символами, равными `.c`. `echo prog.?` — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` `[a-z]` — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

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

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов



проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет Вам возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.

4. Какие операторы используются для прерывания цикла?

Прерывание цикла: операторы `break`, `Continue`, `return` и функция `Abort`.

5. Для чего нужны команды `false` и `true`?

`true`,: - всегда возвращает 0 в качестве кода выхода. `false` - всегда возвращает 1 в качестве кода выхода.

6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?

Введенная строка означает условие существования файла `mans/i.$s`

7. Объясните различия между конструкциями `while` и `until`.

Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.

## 5 Выводы

Я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.