

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

## № 11

Программирование в командном процессоре ОС UNIX. Ветвления и  
циклы

Коняева Марина Александровна

# Содержание

Цель работы	3
Задание	4
Теоретическое введение	5
Выполнение лабораторной работы	6
Выводы	12
Контрольные вопросы	13

## Цель работы

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

# Задание

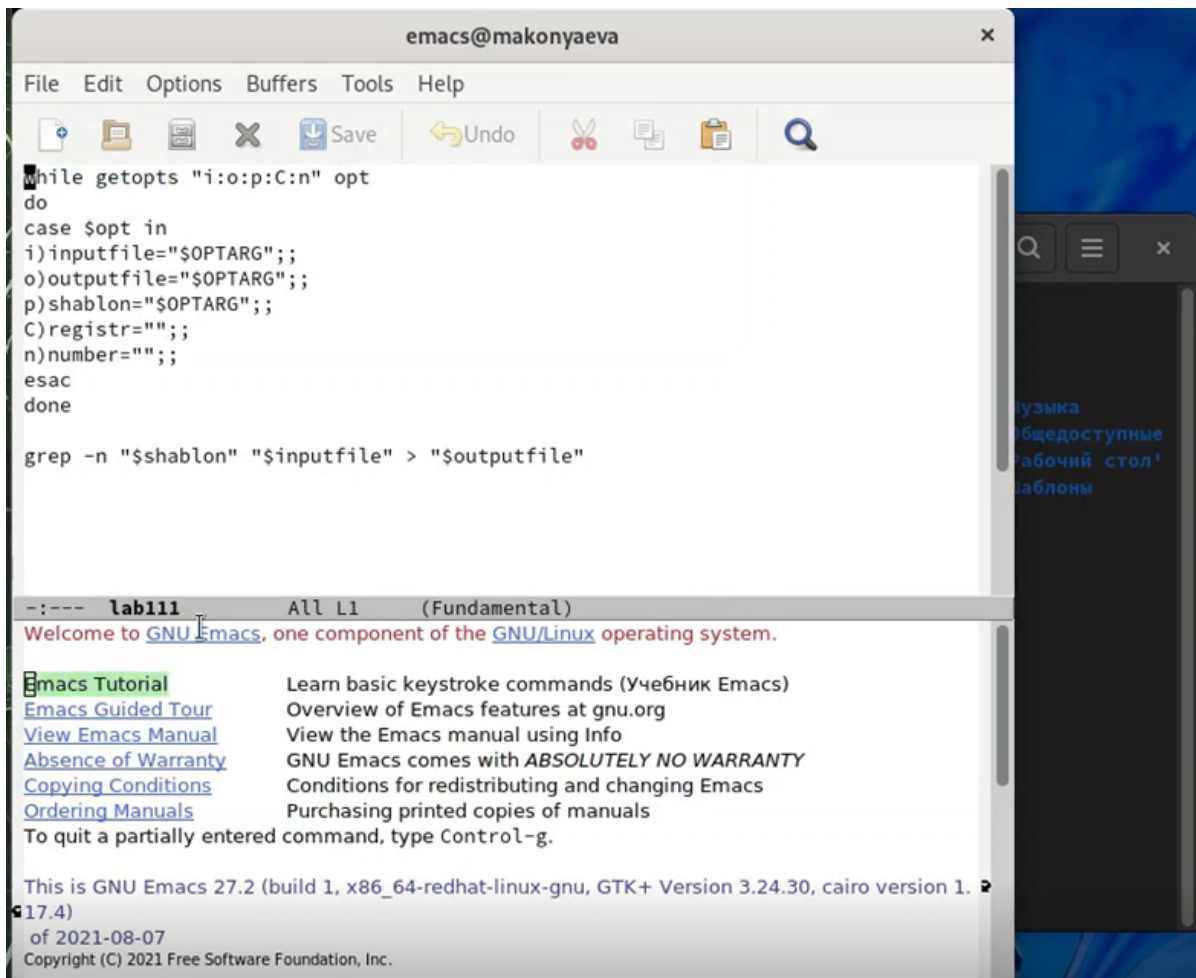
1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами, а затем ищет в указанном файле нужные строки, определяемые ключом `-p`.
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit`, 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 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`).

# Теоретическое введение

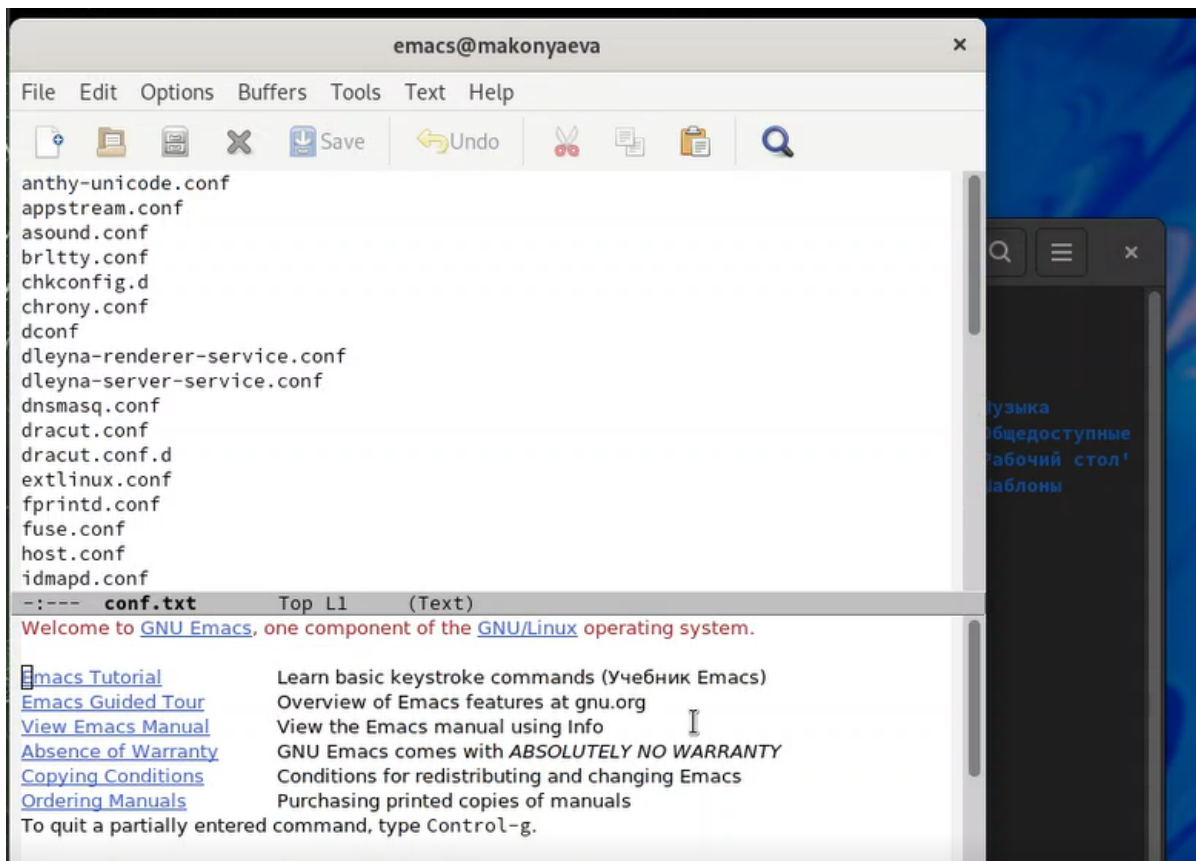
Управление ходом исполнения – один из ключевых моментов структурной организации сценариев на языке командной оболочки. Циклы и преходы являются теми инструментальными средствами, которые обеспечивают управление порядком исполнения команд.

# Выполнение лабораторной работы

## 1. Скрипт 1 (изображение 1.1-2.1)

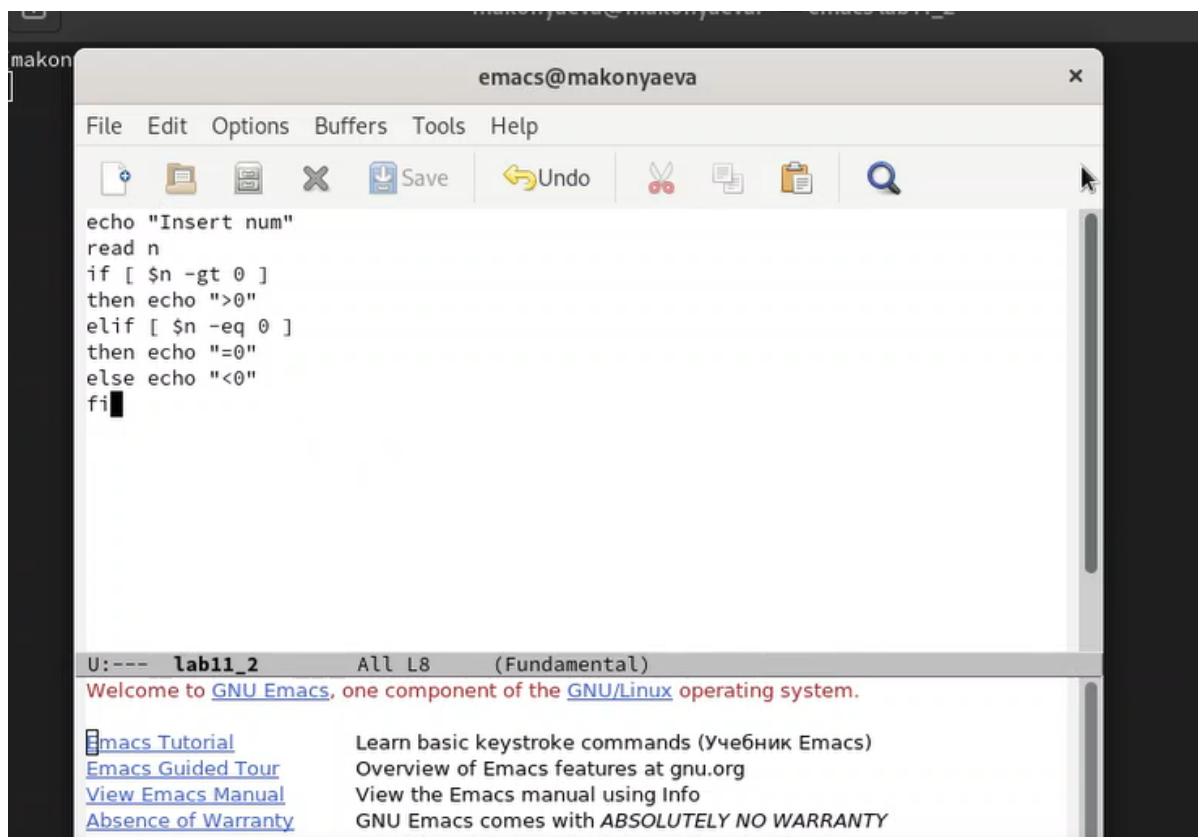


Изображение 1.1 Скрипт 1



Изображение 2.1 Скрипт 1

2. Скрипт 2 (изображение 2.1-2.2)



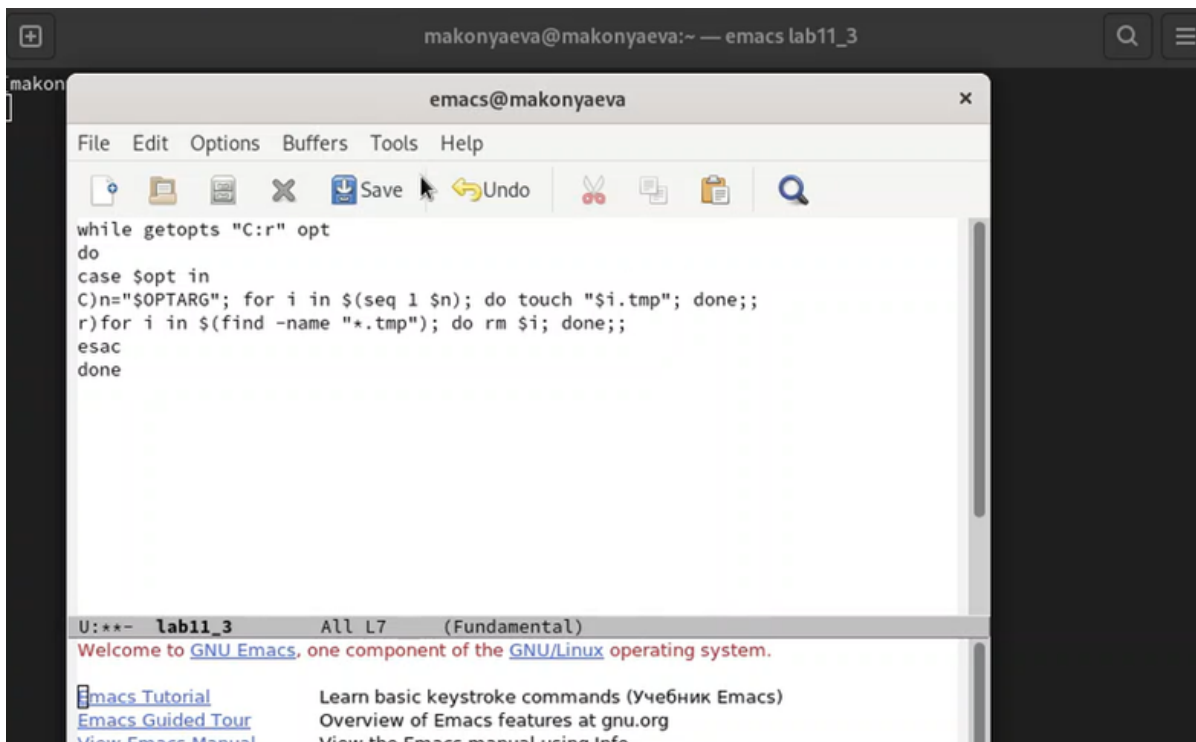
Изображение 2.1 Скрипт 2



```
[makonyaeva@makonyaeva ~]$ chmod +x lab11_2
[makonyaeva@makonyaeva ~]$ ./lab11_2
Insert num
2
>0
[makonyaeva@makonyaeva ~]$ ./lab11_2
Insert num
0
=0
[makonyaeva@makonyaeva ~]$ ./lab11_2
Insert num
-3
<0
[makonyaeva@makonyaeva ~]$
```

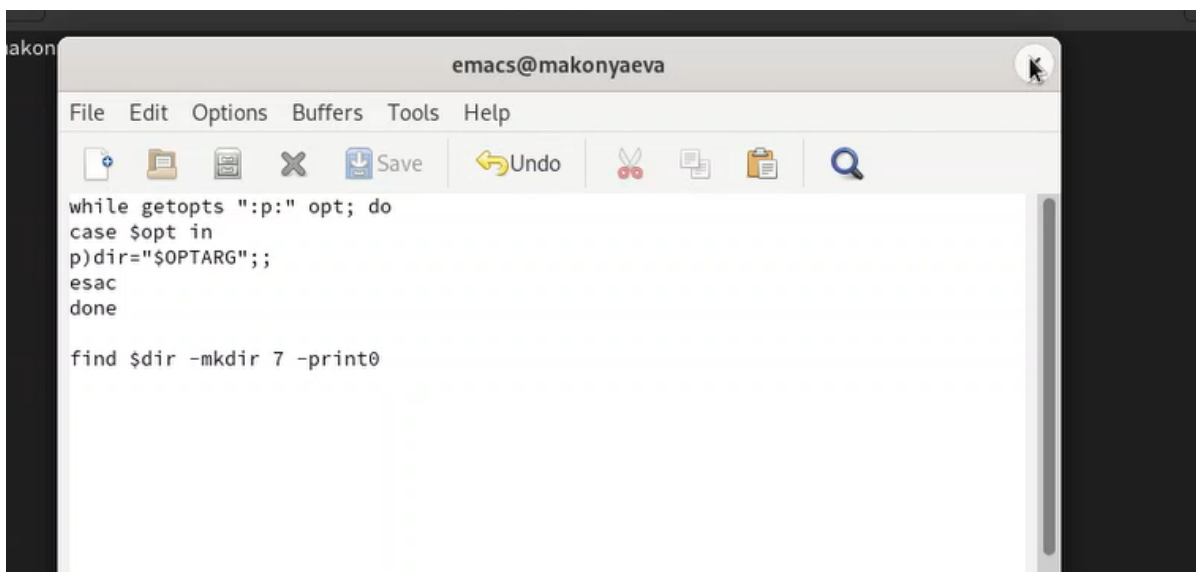
Изображение 2.2 Скрипт 2

### 3. Скрипт 3 (изображение 3.1)

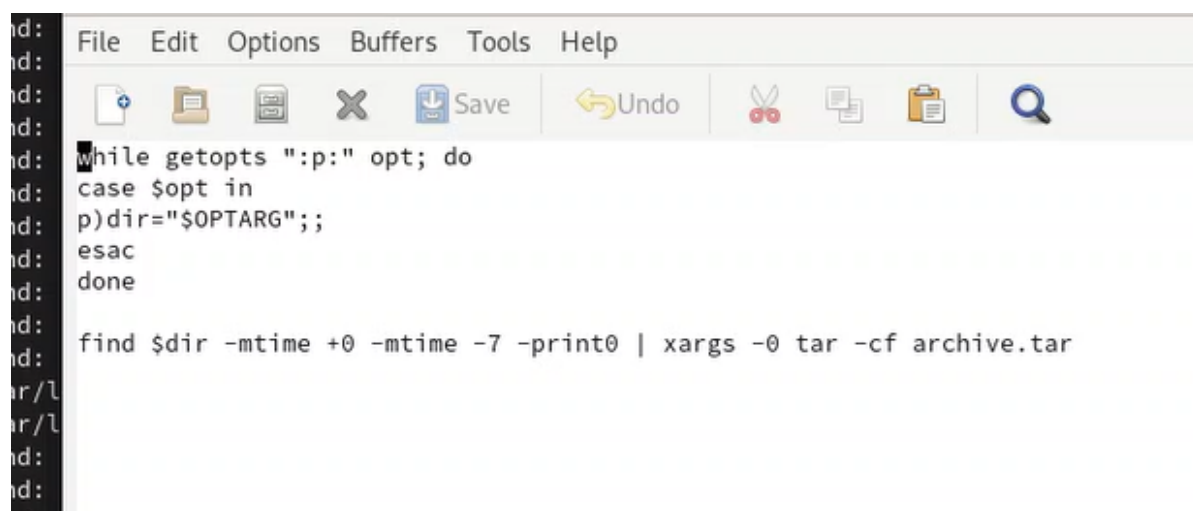


Изображение 3.1 Скрипт 3

#### 4. Скрипт 4 (изображение 4.1-4.2)



Изображение 4.1 Скрипт 4



```
nd: while getopts ":p:" opt; do
nd: case $opt in
nd: p)dir="$OPTARG";;
nd: esac
nd: done
nd:
nd: find $dir -mtime +0 -mtime -7 -print0 | xargs -0 tar -cf archive.tar
ar/l
ar/l
nd:
nd:
```

Изображение 4.2 Скрипт 4

## Выводы

В ходе данной лабораторной работы изучили основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов., а также ответили на контрольные вопросы.

# Контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ... ]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -infile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: 

```
while getopts o:i:Ltr optletter do
case
optletter in
o) oflag = 1; oval =OPTARG;;
i) iflag=1; ival=$OPTARG;;
L) Lflag=1;;
t) tflag=1;;
r) rflag=1;;
*) echo Illegal option $optletter
esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTID`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTID` является числовым индексом на упомянутый аргумент. Функция

getopts также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. 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. 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. Два несложных способа позволяют вам прерывать циклы в оболочке bash. Команда break завершает выполнение цикла, а команда continue завершает данную итерацию блока операторов. Команда break полезна для завершения цикла while в ситуациях, когда условие перестает быть правильным. Пример

бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать: `while true do if [! -f $file] then break fi sleep 10 done`

5. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
6. Введенная строка означает условие существования файла `man s/i.$s`
7. 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`.