

# Лабораторная работа №11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

---

Рогожина Н.А.

13 апреля 2023

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Рогожина Надежда Александровна
- Студентка 1го курса, НКАбд-02-22
- Компьютерные и информационные науки
- Российский университет дружбы народов
- Github

## Вводная часть

---

- Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

- Командный процессор

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

## Задание

---



## Задачи:

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;
- `-ршаблон` — указать шаблон для поиска;
- `-C` — различать большие и малые буквы;
- `-n` — выдавать номера строк.

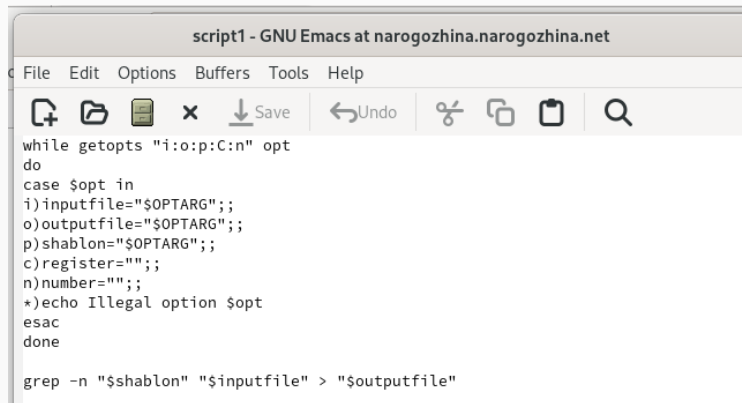
а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.

## Задачи:

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 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`).

## Содержание лабораторной работы

---



```
script1 - GNU Emacs at narogozhina.narogozhina.net
File Edit Options Buffers Tools Help
[Icons: Open, Save, Undo, Redo, Search]

while getopts "i:o:p:C:n" opt
do
case $opt in
i)inputfile="$OPTARG";;
o)outputfile="$OPTARG";;
p)shablon="$OPTARG";;
c)register="";;
n)number="";;
*)echo Illegal option $opt
esac
done

grep -n "$shablon" "$inputfile" > "$outputfile"
```

Рис. 1: Скрипт №1

## Запуск первого скрипта

```
[narogozhina@narogozhina ~]$ emacs script1
[narogozhina@narogozhina ~]$ chmod +x script1
[narogozhina@narogozhina ~]$ ./script1 -i conf.txt -o output.txt -pm etconf -C -n
./script1: строка 1: getops: команда не найдена
./script1: строка 13: : Нет такого файла или каталога
[narogozhina@narogozhina ~]$ emacs script1
[narogozhina@narogozhina ~]$ ./script1 -i conf.txt -o output.txt -pm etconf -C -n
[narogozhina@narogozhina ~]$ emacs script1
[narogozhina@narogozhina ~]$ ./script1 -i conf.txt -o output.txt -pm etconf -C -n
```

Рис. 2: Запуск первого скрипта

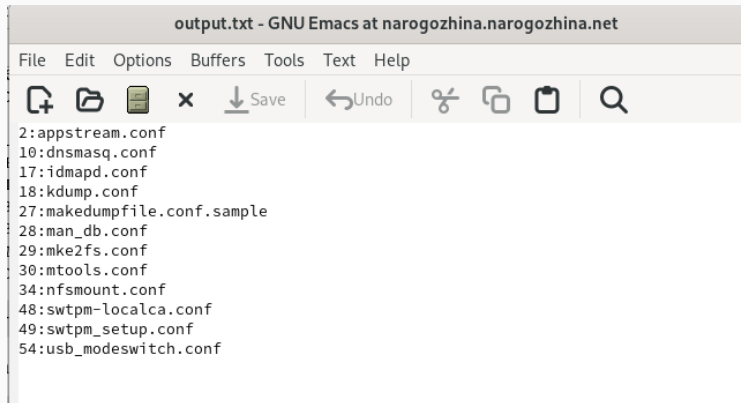
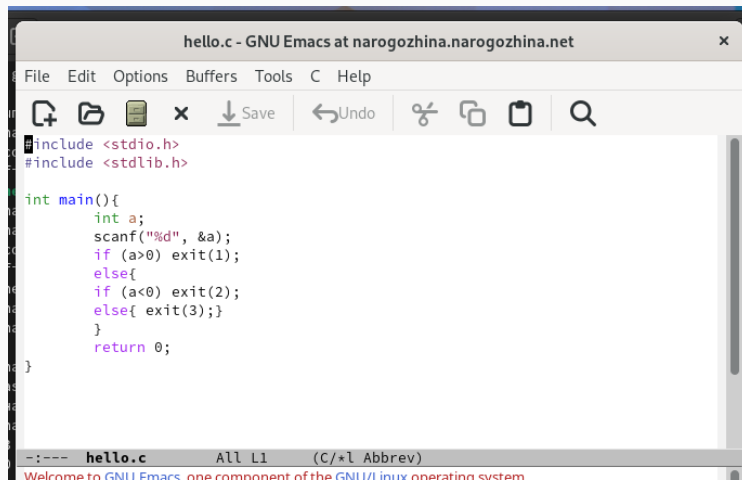


Рис. 3: Результат

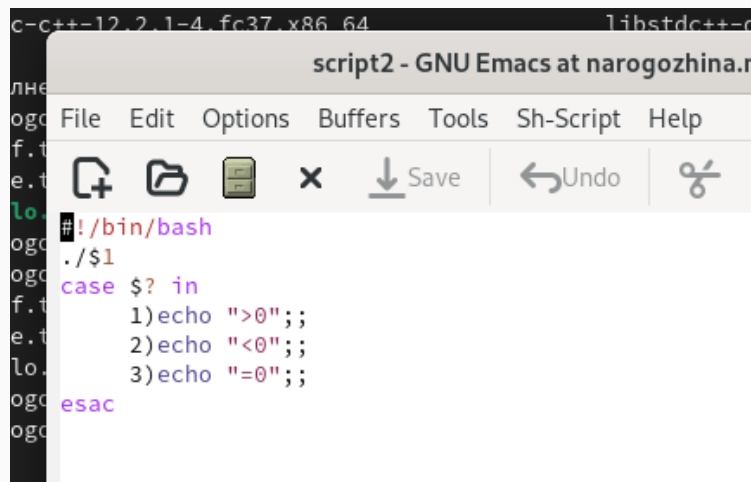


```
hello.c - GNU Emacs at narogozhina.narogozhina.net
File Edit Options Buffers Tools C Help
[Icons] Save Undo [Icons] [Icons] [Icons]
#include <stdio.h>
#include <stdlib.h>

int main(){
    int a;
    scanf("%d", &a);
    if (a>0) exit(1);
    else{
        if (a<0) exit(2);
        else{ exit(3);}
    }
    return 0;
}

--- hello.c All L1 (C/*l Abbrev)
Welcome to GNU Emacs, one component of the GNU/Linux operating system
```

Рис. 4: Программа на C



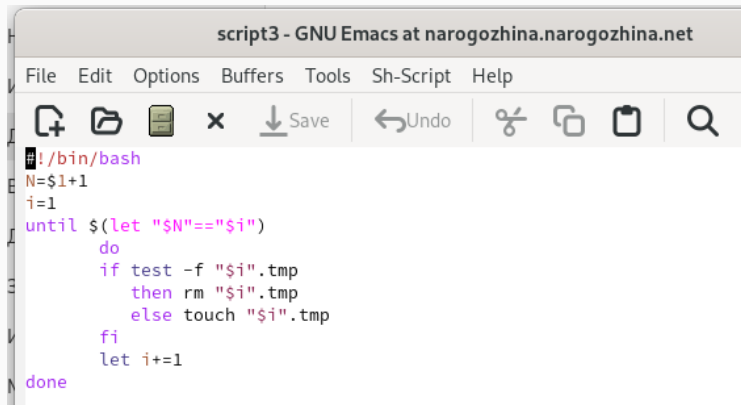
```
c-c++-12.2.1-4.fc37.x86_64 libstdc++-c
script2 - GNU Emacs at narogozhina.r
File Edit Options Buffers Tools Sh-Script Help
[Icons: Open, Save, Undo, Scissors]
#!/bin/bash
./$1
case $? in
    1)echo ">0";;
    2)echo "<0";;
    3)echo "=0";;
esac
```

Рис. 5: Скрипт №2



```
[narogozhina@narogozhina ~]$ bash script2 hello
5
>0
[narogozhina@narogozhina ~]$ bash script2 hello
0
=0
[narogozhina@narogozhina ~]$ bash script2 hello
-6
<0
[narogozhina@narogozhina ~]$
```

Рис. 6: Выполнение скрипта №2



```
#!/bin/bash
N=$1+1
i=1
until $(let "$N"=="$i")
do
    if test -f "$i".tmp
    then rm "$i".tmp
    else touch "$i".tmp
    fi
    let i+=1
done
```

Рис. 7: Скрипт №3

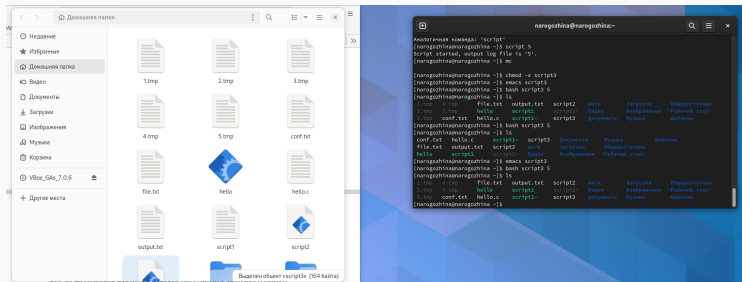


Рис. 8: Выполнение скрипта №3

# Выполнение повторно

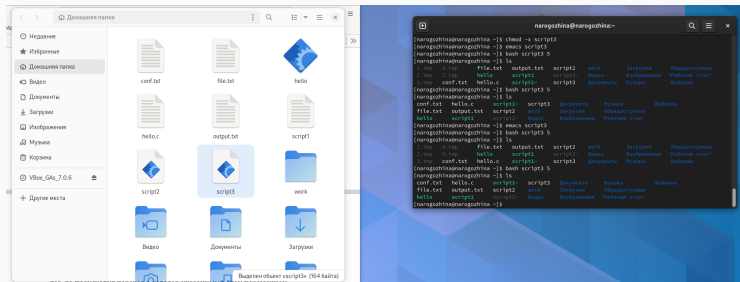
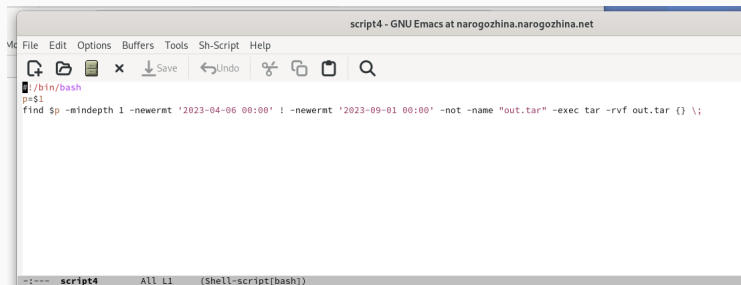


Рис. 9: Повторное выполнение скрипта №3



```
script4 - GNU Emacs at narogzhina.narogzhina.net
File Edit Options Buffers Tools Sh-Script Help
[Icons: Open, Save, Undo, Cut, Copy, Paste, Find]
~/bin/bash
p=$1
find $p -mindepth 1 -newermt '2023-04-06 00:00' ! -newermt '2023-09-01 00:00' -not -name "out.tar" -exec tar -rvf out.tar {} \;
```

~:-- script4 All L1 (Shell-script[bash])

Рис. 10: Скрипт №4

# Пример выполнения

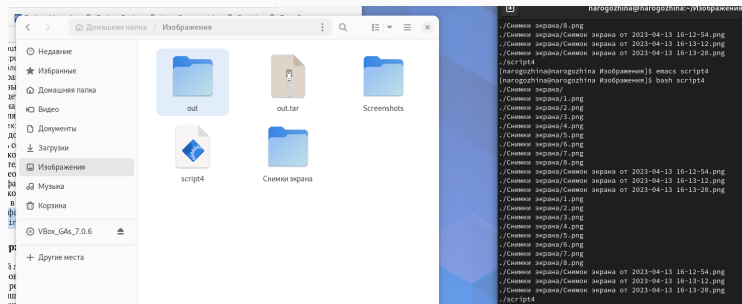


Рис. 11: Запуск скрипта

Здесь представлен архив out, который был создан во время работы скрипта, и в нем:

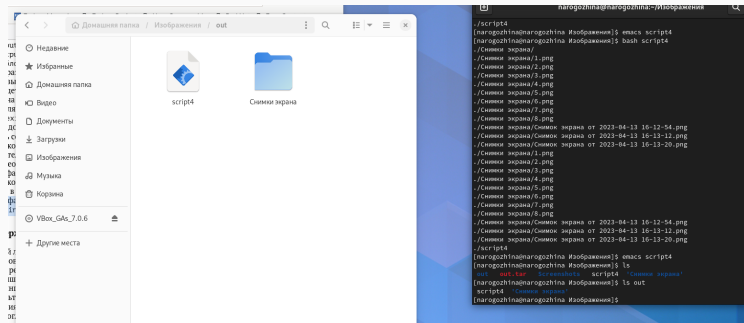


Рис. 12: Папка out

## Выводы

---



В ходе лабораторной работы мы научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.