

# Лабораторная работа №13

Программирование в командном процессоре ОС UNIX. Ветвления и циклы.

---

Головина М.И.

10 мая 2025

Российский университет дружбы народов, Москва, Россия

Факультет Физико-математических и естественных наук

## Информация

---

- Головина Мария Игоревна
- Бакалавр направления подготовки Математика и механика
- студентка группы НММбд - 02- 24
- Российский университет дружбы народов
- 1132246810@pfur.ru



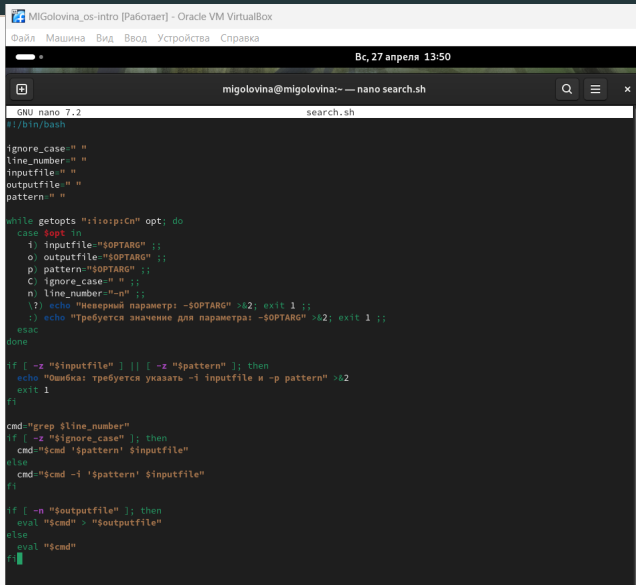
- Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку, а затем ищет в указанном файле нужные строки.
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.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N.
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.
5. Ответить на контрольные вопросы.

## Ход работы

---

Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку, а затем ищет в указанном файле нужные строки



The screenshot shows a terminal window titled "MIGolovina\_os-intro [Работаer] - Oracle VM VirtualBox". The window contains a nano editor editing a file named "search.sh". The script is written in Bash and uses `getopts` to parse command-line options. It defines variables for `inputfile`, `outputfile`, `pattern`, and `ignore_case`. It then uses `grep` to search for a pattern in a file, with options for case sensitivity and line numbers. The script includes error handling for missing arguments and invalid options.

```
#!/bin/bash

ignore_case=""
line_number=""
inputfile=""
outputfile=""
pattern=""

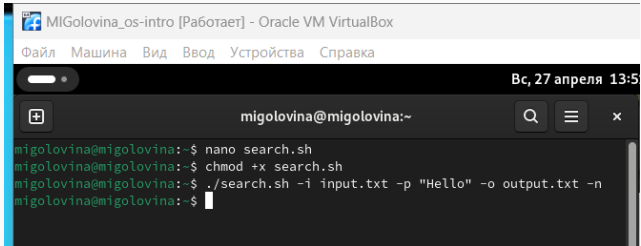
while getopts ":i:o:p:Cn" opt; do
  case $opt in
    i) inputfile="$OPTARG" ;;
    o) outputfile="$OPTARG" ;;
    p) pattern="$OPTARG" ;;
    C) ignore_case="" ;;
    n) line_number="-n" ;;
    \?) echo "Неверный параметр: -$OPTARG" >&2; exit 1 ;;
    :) echo "Требуется значение для параметра: -$OPTARG" >&2; exit 1 ;;
  esac
done

if [ -z "$inputfile" ] || [ -z "$pattern" ]; then
  echo "Ошибка: требуется указать -i inputfile и -p pattern" >&2
  exit 1
fi

cmd="grep $line_number"
if [ -z "$ignore_case" ]; then
  cmd="$cmd '$pattern' $inputfile"
else
  cmd="$cmd -i '$pattern' $inputfile"
fi

if [ -n "$outputfile" ]; then
  eval "$cmd" > "$outputfile"
else
  eval "$cmd"
fi
```

Запустила скрипт №1



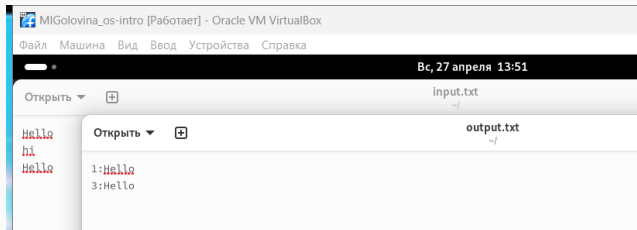
The screenshot shows a terminal window titled "MIGolovina\_os-intro [Работает] - Oracle VM VirtualBox". The window has a menu bar with "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The terminal prompt is "migolovina@migolovina:~". The commands entered are:

```
migolovina@migolovina:~$ nano search.sh
migolovina@migolovina:~$ chmod +x search.sh
migolovina@migolovina:~$ ./search.sh -i input.txt -p "Hello" -o output.txt -n
migolovina@migolovina:~$
```

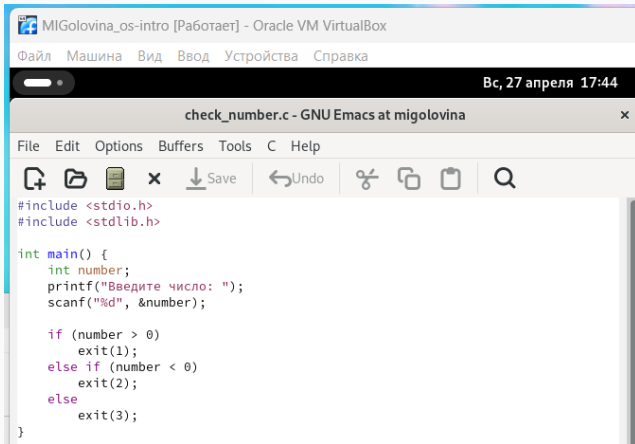
The terminal window also shows a date and time in the top right corner: "Вс, 27 апреля 13:5".



Проверила правильность работы  
скрипта №1



Написала на языке Си программу,  
которая вводит число и определяет,  
является ли оно больше нуля,  
меньше нуля или равно нулю

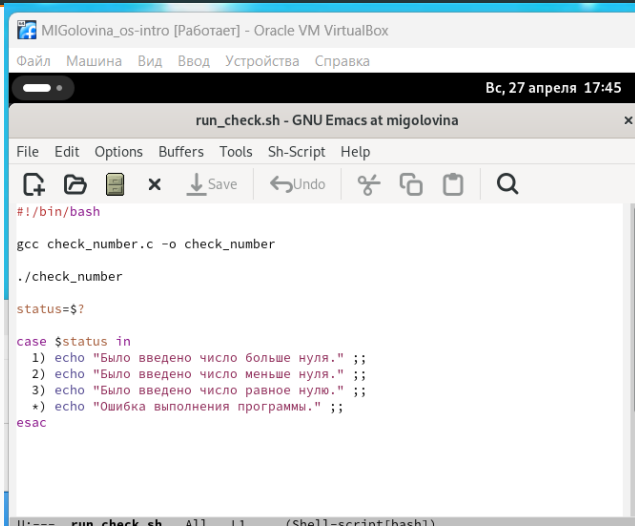
A screenshot of a virtual machine window titled "MIGolovina\_os-intro [Работает] - Oracle VM VirtualBox". Inside the VM, the GNU Emacs editor is open with a file named "check\_number.c". The menu bar shows "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The status bar at the top right indicates the date and time: "Вс, 27 апреля 17:44". The Emacs interface includes a menu bar with "File", "Edit", "Options", "Buffers", "Tools", "C", and "Help". Below the menu bar is a toolbar with icons for file operations and editing. The main text area contains the following C code:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int number;
    printf("Введите число: ");
    scanf("%d", &number);

    if (number > 0)
        exit(1);
    else if (number < 0)
        exit(2);
    else
        exit(3);
}
```

Затем написала программу, которая завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку



```
#!/bin/bash

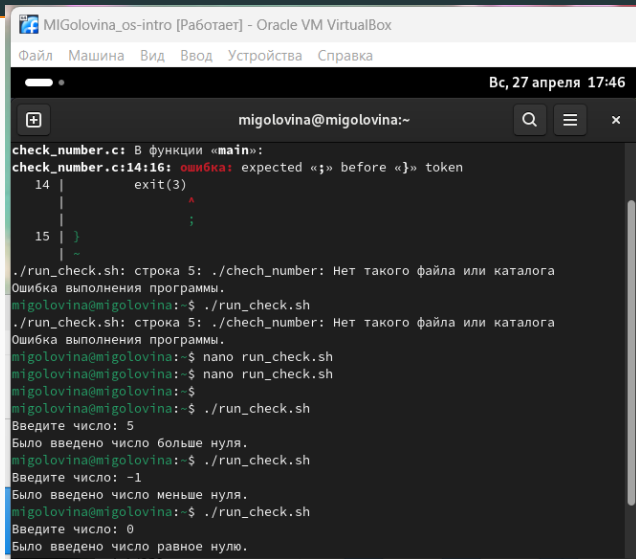
gcc check_number.c -o check_number

./check_number

status=$?

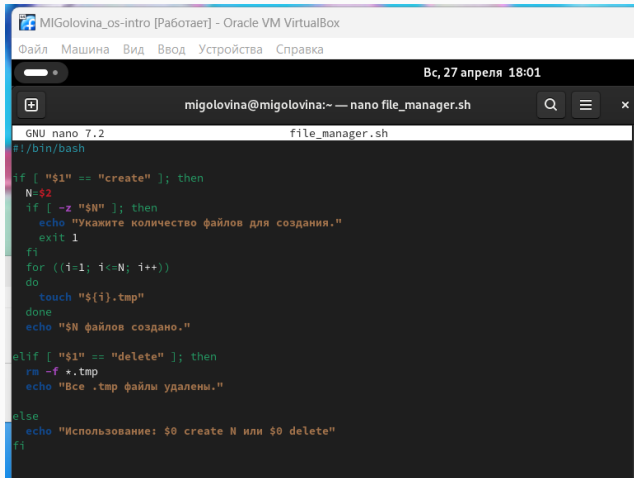
case $status in
  1) echo "Было введено число больше нуля." ;;
  2) echo "Было введено число меньше нуля." ;;
  3) echo "Было введено число равное нулю." ;;
  *) echo "Ошибка выполнения программы." ;;
esac
```

Запустила скрипт №2



```
MIGolovina_os-intro [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 27 апреля 17:46
migolovina@migolovina:~
check_number.c: В функции «main»:
check_number.c:14:16: ошибка: expected «;» before «}» token
14 |         exit(3)
    |                ^
15 |     }
    |     ~
./run_check.sh: строка 5: ./chech_number: Нет такого файла или каталога
Ошибка выполнения программы.
migolovina@migolovina:~$ ./run_check.sh
./run_check.sh: строка 5: ./chech_number: Нет такого файла или каталога
Ошибка выполнения программы.
migolovina@migolovina:~$ nano run_check.sh
migolovina@migolovina:~$ nano run_check.sh
migolovina@migolovina:~$ ./run_check.sh
Введите число: 5
Было введено число больше нуля.
migolovina@migolovina:~$ ./run_check.sh
Введите число: -1
Было введено число меньше нуля.
migolovina@migolovina:~$ ./run_check.sh
Введите число: 0
Было введено число равное нулю.
```

Написала командный файл,  
создающий указанное число файлов,  
пронумерованных последовательно  
от 1 до N



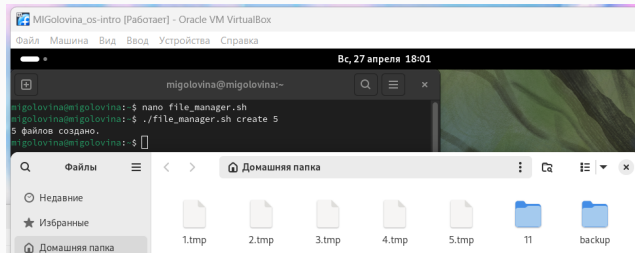
```
MIGolovina_os-intro [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 27 апреля 18:01
migolovina@migolovina:~ — nano file_manager.sh
GNU nano 7.2                                file_manager.sh
#!/bin/bash

if [ "$1" == "create" ]; then
N=$2
if [ -z "$N" ]; then
echo "Укажите количество файлов для создания."
exit 1
fi
for ((i=1; i<=N; i++))
do
touch "${i}.tmp"
done
echo "$N файлов создано."

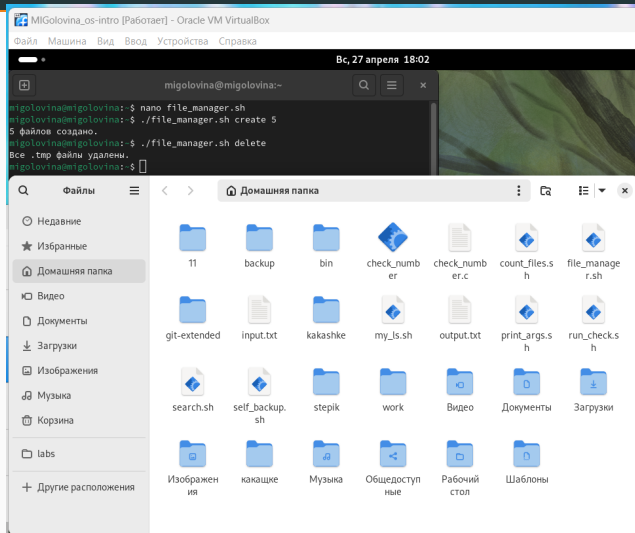
elif [ "$1" == "delete" ]; then
rm -f *.tmp
echo "Все .tmp файлы удалены."

else
echo "Использование: $0 create N или $0 delete"
fi
```

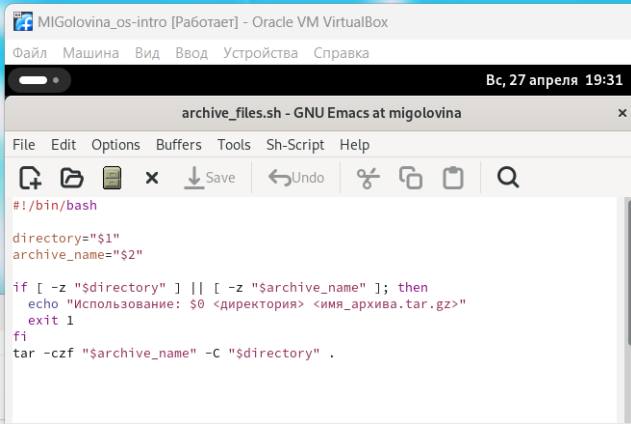
Запустила скрипт №3 с созданием  
файлов



Запустила скрипт №3 с удалением  
файлов



Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории



```
#!/bin/bash

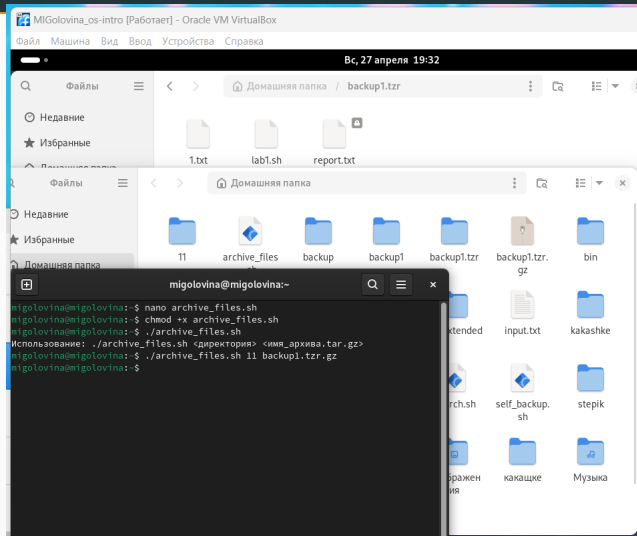
directory="$1"
archive_name="$2"

if [ -z "$directory" ] || [ -z "$archive_name" ]; then
    echo "Использование: $0 <директория> <имя_архива.tar.gz>"
    exit 1
fi

tar -czf "$archive_name" -C "$directory" .
```

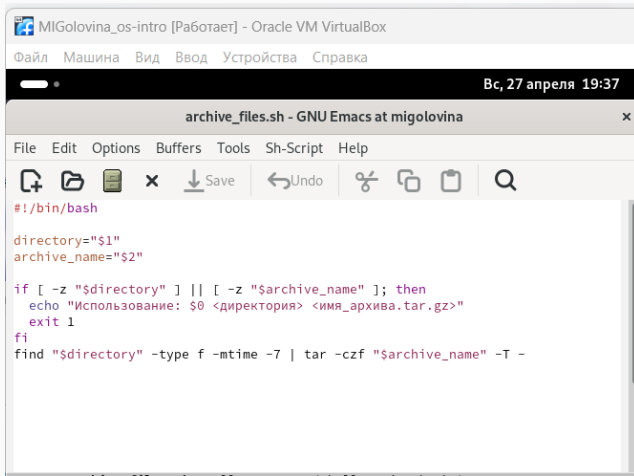


Проверила правильность работы  
скрипта №4



## Модифицированный скрипт №4

Модифицировала его так, чтобы  
запаковывались только те файлы,  
которые были изменены менее  
недели тому назад



The screenshot shows a virtual machine window titled "MIGolovina\_os-intro [Работает] - Oracle VM VirtualBox". Inside the VM, a GNU Emacs editor window titled "archive\_files.sh - GNU Emacs at migolovina" is open. The editor displays a shell script with the following content:

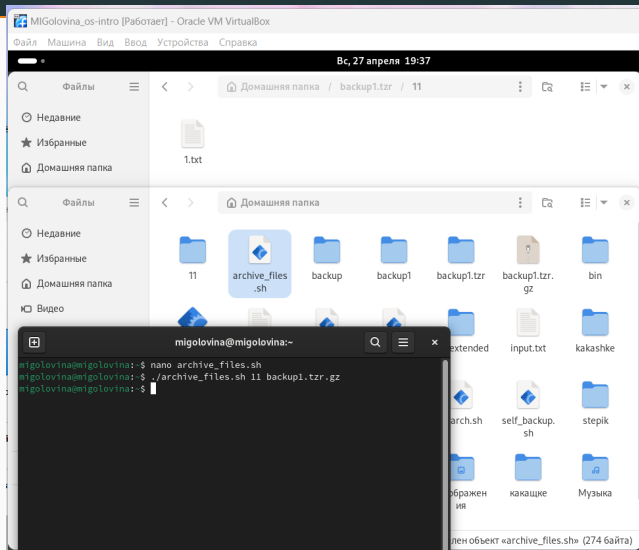
```
#!/bin/bash

directory="$1"
archive_name="$2"

if [ -z "$directory" ] || [ -z "$archive_name" ]; then
    echo "Использование: $0 <директория> <имя_архива.tar.gz>"
    exit 1
fi

find "$directory" -type f -mtime -7 | tar -czf "$archive_name" -T -
```

Проверила правильность работы  
модифицированного скрипта №4



## Ответы на контрольные вопросы

---

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

Команда `getopts` в Bash используется для разбора и обработки опций (аргументов) командной строки, переданных скрипту или функции. Она позволяет удобно управлять флагами и параметрами, которые пользователь может указать при запуске скрипта.

Вот основные аспекты предназначения и использования `getopts`:

1. Обработка опций `getopts` позволяет обрабатывать короткие (один символ) и длинные (более одного символа) опции, что делает интерфейс вашего скрипта более удобным и понятным.

### 2. Синтаксис

Основной синтаксис команды выглядит следующим образом:

```
getopts "options" variable
```

## Вывод

---

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

Дорогу осилит идущий

---