

# **Отчет по лабораторной работе №13**

**Программирование в командном процессоре ОС UNIX. Командные  
файлы**

Чигладзе Майя Владиславовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Порядок выполнения лабораторной работы</b>	<b>6</b>
2.1	Задание 1. . . . .	6
2.2	Задание 2. . . . .	7
2.3	Задание 3. . . . .	8
2.4	Задание 4. . . . .	10
<b>3</b>	<b>Ответы на контрольные вопросы</b>	<b>11</b>
<b>4</b>	<b>Выводы</b>	<b>12</b>
	<b>Список литературы</b>	<b>13</b>

## Список иллюстраций

2.1	Код . . . . .	6
2.2	Код ч.1 . . . . .	7
2.3	Код ч.2 . . . . .	8
2.4	Результат . . . . .	8
2.5	Код . . . . .	9
2.6	Результат . . . . .	9
2.7	Код . . . . .	10
2.8	Результат . . . . .	10

## 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 Порядок выполнения лабораторной работы

### 2.1 Задание 1.

Задача: 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; `-r`шаблон — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк

#### 1. Код/файл (рис. 1).

```
root@mvchigladze:/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/lab13# nano first.md
root@mvchigladze:/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/lab13# grep -i -o -p -C -n 'Maia' first.md first_output.md
grep: неверный ключ - «p»
Использование: grep [ПАРАМЕТР]... ШАБЛОНЫ [ФАЙЛ]...
Запустите «grep --help» для получения более подробного описания.
root@mvchigladze:/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/lab13# grep -i -o -p -C -n 'Maia' first.md first_output.md
grep: неверный ключ - «p»
Использование: grep [ПАРАМЕТР]... ШАБЛОНЫ [ФАЙЛ]...
Запустите «grep --help» для получения более подробного описания.
root@mvchigladze:/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/lab13# grep -i -o -C -n 'Maia' first.md first_output.md
grep: -n: неверный аргумент длины контекста
root@mvchigladze:/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/lab13# grep -i -o -p 'Maia' first.md first_output.md
grep: неверный ключ - «p»
Использование: grep [ПАРАМЕТР]... ШАБЛОНЫ [ФАЙЛ]...
Запустите «grep --help» для получения более подробного описания.
root@mvchigladze:/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/lab13# ls
first.md
```

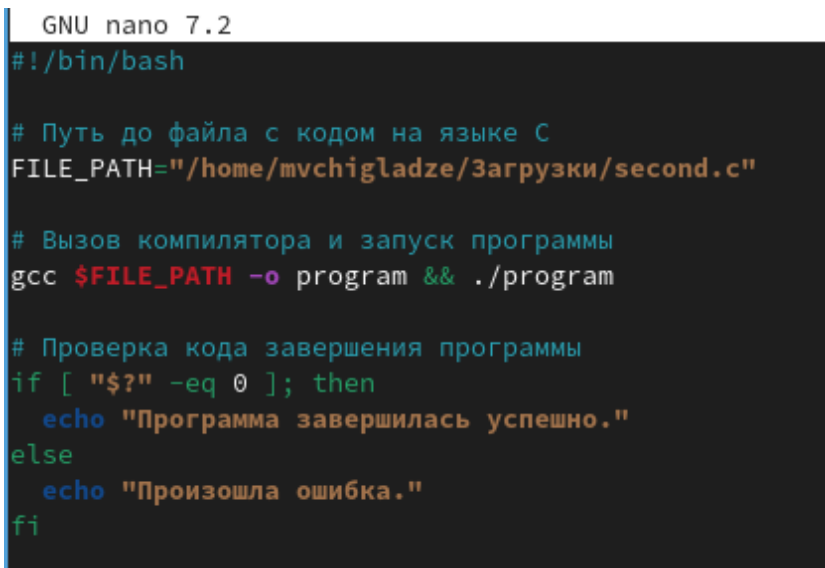
Рис. 2.1: Код

## 2.2 Задание 2.

Задача: Записать 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.

Эта программа на языке C принимает число от пользователя, а затем определяет, больше ли это число нуля, меньше или равно. После этого программа завершается, возвращая значение 0, что означает успешное выполнение.

1. Код/файл первый (рис. 2).

A screenshot of a terminal window with a dark background. At the top, it says 'GNU nano 7.2'. The script content is as follows:

```
#!/bin/bash

# Путь до файла с кодом на языке C
FILE_PATH="/home/mvchigladze/Загрузки/second.c"

# Вызов компилятора и запуск программы
gcc $FILE_PATH -o program && ./program

# Проверка кода завершения программы
if [ "$?" -eq 0 ]; then
    echo "Программа завершилась успешно."
else
    echo "Произошла ошибка."
fi
```

Рис. 2.2: Код ч.1

2. Код/файл второй (рис. 3).

```

GNU nano 7.2
#include <stdio.h>
int main() {
int number;
printf("Введите число: ");
scanf("%d", &number);
if (number > 0) {
    printf("Число больше нуля.\n");
} else if (number < 0) {
    printf("Число меньше нуля.\n");
} else {
    printf("Число равно нулю.\n");
}

return 0;
}

```

Рис. 2.3: Код ч.2

3. Результат (рис. 4).

```

root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab13# ./second.sh
Введите число: 14
Число больше нуля.
Программа завершилась успешно.

```

Рис. 2.4: Результат

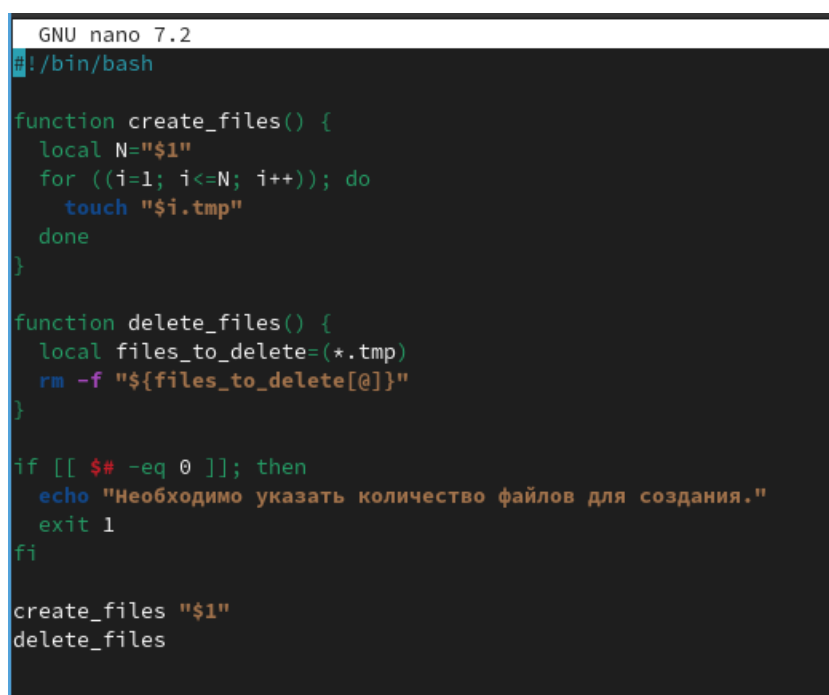
## 2.3 Задание 3.

Задача: Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до  $\square$  (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).



Этот командный файл сначала проверяет, было ли указано количество файлов для создания. Если нет, он выдает сообщение об ошибке и завершает работу. Затем он вызывает функцию `create_files`, передавая ей количество файлов в качестве аргумента. Эта функция создает файлы с номерами от 1 до указанного количества.

1. Код/файл (рис. 5).



```
GNU nano 7.2
#!/bin/bash

function create_files() {
    local N="$1"
    for ((i=1; i<=N; i++)); do
        touch "$i.tmp"
    done
}

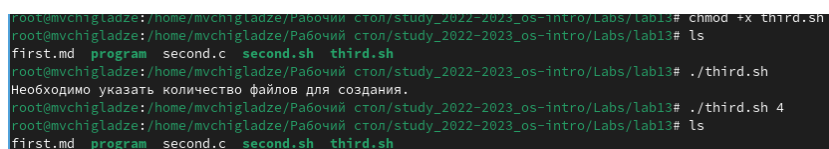
function delete_files() {
    local files_to_delete=(*.tmp)
    rm -f "${files_to_delete[@]}"
}

if [[ $# -eq 0 ]]; then
    echo "Необходимо указать количество файлов для создания."
    exit 1
fi

create_files "$1"
delete_files
```

Рис. 2.5: Код

2. Результат (рис. 6).



```
root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab13# chmod +x third.sh
root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab13# ls
first.md  program  second.c  second.sh  third.sh
root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab13# ./third.sh
Необходимо указать количество файлов для создания.
root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab13# ./third.sh 4
root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab13# ls
first.md  program  second.c  second.sh  third.sh
```

Рис. 2.6: Результат

## 2.4 Задание 4.

Задача: 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).

Этот командный файл принимает два аргумента: директорию, в которой необходимо найти файлы, и имя создаваемого архива. Он использует команду find для поиска файлов, измененных меньше недели назад, и передает их список команде tar для создания архива. Затем он удаляет исходные файлы.

### 1. Код/файл (рис. 7).

```
GNU nano 7.2
# Командный файл для выполнения операций с использованием grep, tar и find

# Аргументы командной строки: директория, в которой нужно найти файлы, и имя архива
if [ -z "$1" ] || [ -z "$2" ]; then
    echo "Использование: `basename $0` директория архив"
    exit 1
fi

# Поиск файлов, измененных менее недели назад
find "$1" -newermt '7 days ago' -print0 |

# Передача списка файлов в tar для создания архива
xargs -0 tar -cvzf "$2".tar.gz

# Удаление исходных файлов
find "$1" -newermt '7 days ago' -exec rm -f {} +
```

Рис. 2.7: Код

### 2. Результат (рис. 8).

```
root@mvchigladze:/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13# ./thourth.sh /home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13#
tar: Удаляется начальный '/' из имен объектов
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/
tar: Удаляется начальное '/' из цепей жестких ссылок
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/first.md
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/second.c
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/second.sh
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/program
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/third.sh
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/thourth.sh
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/first.md
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/second.c
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/second.sh
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/program
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/third.sh
/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13/thourth.sh
rm: невозможно удалить '/home/mvchigladze/Рабочий_стол/study_2022-2023_os-intro/Labs/Lab13': Это каталог
```

Рис. 2.8: Результат

### 3 Ответы на контрольные вопросы

1. Команда `getopts` предназначена для обработки опций и аргументов в Unix-подобных операционных системах.
2. Метасимволы используются при генерации имен файлов для обозначения специальных символов или классов символов.
3. Операторы управления действиями включают условные операторы (`if-else`), циклы (`for`, `while`, `until`), операторы перехода (`break`, `continue`).
4. Операторы для прерывания циклов включают `break` и `continue`.
5. Команды `false` и `true` используются в Unix-системах для возврата кода ошибки (`false`) или успешного выполнения (`true`).
6. Строка `if test -f mans/i.s` проверяет, является ли файл `mans/i.s` существующим и доступным для чтения. Если условие выполняется, то выполняется следующая команда.
7. Конструкция `while` выполняет блок кода до тех пор, пока заданное условие истинно, в то время как конструкция `until` выполняет код до тех пор, пока условие не станет истинным. Основное различие между ними заключается в том, что `while` проверяет условие перед выполнением блока кода, а `until` проверяет условие после выполнения блока кода.

## **4 Выводы**

В ходе лабораторной работы, я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## **Список литературы**