

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №11**

*дисциплина: Операционные системы*

Шилоносов Данил Вячеславович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задачи</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Задание №1 . . . . .	7
3.2	Задание №2 . . . . .	9
3.3	Задание №3 . . . . .	10
3.4	Задание №4 . . . . .	12
<b>4</b>	<b>Выводы</b>	<b>14</b>

## Список иллюстраций

3.1	Исходный код скрипта №1 . . . . .	8
3.2	Результат выполнения скрипта №1 . . . . .	8
3.3	Исходный код скрипта №2 . . . . .	9
3.4	Результат выполнения скрипта №2 . . . . .	10
3.5	Исходный код скрипта №3 . . . . .	11
3.6	Результат выполнения скрипта №3 . . . . .	11
3.7	Исходный код скрипта №4 . . . . .	12
3.8	Результат выполнения скрипта №4 . . . . .	13

# 1 Цель работы

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

## 2 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;
  - `-С` — различать большие и малые буквы;
  - `-п` — 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 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).

## 3 Выполнение лабораторной работы

### 3.1 Задание №1

1. Данный командный файл на Bash обрабатывает параметры командной строки и выполняет поиск строк в указанном файле на основе заданного шаблона. Он поддерживает следующие параметры:

- -i inputfile - указывает входной файл для чтения данных.
- -o outputfile - указывает выходной файл для записи результатов.
- -r pattern - указывает шаблон для поиска в файле.
- -C - различает большие и малые буквы при поиске.
- -n - выводит номера строк, в которых найден заданный шаблон.

Если параметр -r не указан, скрипт выведет сообщение об ошибке и завершится. Если входной файл не существует, скрипт также выведет сообщение об ошибке и завершится.

После обработки параметров, скрипт настраивает опции для команды grep, используя флаги -i (если -C указан) и -n (если -n указан). Затем скрипт выполняет поиск заданного шаблона в указанном файле, используя команду grep. Результаты выводятся в консоль или в указанный выходной файл в зависимости от того, указан ли параметр -o (рис. [3.1], [3.2])

```
dvshilonosov@dvshilonosov:~/scripts/script1 — vim script.sh

# Проверяем, был ли указан обязательный параметр -p
if [ -z "$pattern" ]; then
    echo "Не указан обязательный параметр -p" >&2
    exit 1
fi

# Проверяем, существует ли указанный файл
if [ ! -f "$inputfile" ]; then
    echo "Файл $inputfile не существует" >&2
    exit 1
fi

# Настраиваем опции для команды grep в зависимости от флагов
grep_options=""
if [ $case_sensitive = true ]; then
    grep_options="$grep_options -i"
fi
if [ $line_numbers = true ]; then
    grep_options="$grep_options -n"
fi

# Выполняем поиск и выводим результаты в файл или на экран
if [ -z "$outputfile" ]; then
    grep $grep_options "$pattern" "$inputfile"
else
    grep $grep_options "$pattern" "$inputfile" > "$outputfile"
fi
```

Рис. 3.1: Исходный код скрипта №1

```
dvshilonosov@dvshilonosov:~/scripts/script1 — vim script.sh
[1] dvshilonosov@dvshilonosov:~/scripts/script1$ bash script.sh -i input.txt -o output.txt -p "hello world" -C -n
2:Hello world Morbi vulputate, nibh at elementum pellentesque, odio metus rhoncus leo,
7:Pellentesque ut risus quis velit congue convallis eu vel leo. Aliquam erat volutpat. Hello world Sed maximus risus eget massa pharetra consequat.
11:Hello world Etiam vitae risus arcu. Sed vel augue et elit bibendum ultricies. Nullam eleifend elit in augue lacinia rhoncus.
14:Hello world
```

Рис. 3.2: Результат выполнения скрипта №1

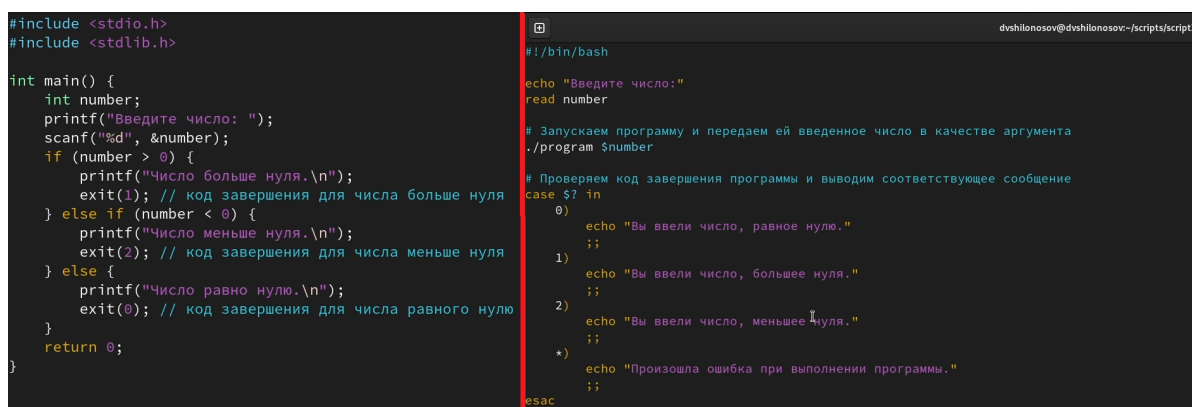


## 3.2 Задание №2

2. Этот скрипт запрашивает у пользователя ввод числа, запускает программу на языке C, передает ей введенное число в качестве аргумента, а затем выводит сообщение в зависимости от кода завершения программы. Затем скрипт выводит на экран приглашение для ввода числа. Когда пользователь вводит число, оно сохраняется в переменной `number`.

Далее скрипт вызывает программу на языке C и передает ей введенное число в качестве аргумента.

После того, как программа завершилась, скрипт проверяет код завершения программы с помощью команды `$?` и выводит соответствующее сообщение, основываясь на коде завершения. Если код завершения равен 0, то выводится сообщение “Вы ввели число, равное нулю.”, если код завершения равен 1, то выводится сообщение “Вы ввели число, большее нуля.”, а если код завершения равен 2, то выводится сообщение “Вы ввели число, меньшее нуля.” (рис. [3.3], [3.4])



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

int main() {
    int number;
    printf("Введите число: ");
    scanf("%d", &number);
    if (number > 0) {
        printf("Число больше нуля.\n");
        exit(1); // код завершения для числа больше нуля
    } else if (number < 0) {
        printf("Число меньше нуля.\n");
        exit(2); // код завершения для числа меньше нуля
    } else {
        printf("Число равно нулю.\n");
        exit(0); // код завершения для числа равного нулю
    }
    return 0;
}
```

```
#!/bin/bash

echo "Введите число:"
read number

# Запускаем программу и передаем ей введенное число в качестве аргумента
./program $number

# Проверяем код завершения программы и выводим соответствующее сообщение
case $? in
    0)    echo "Вы ввели число, равное нулю."
          ;;
    1)    echo "Вы ввели число, большее нуля."
          ;;
    2)    echo "Вы ввели число, меньшее нуля."
          ;;
    *)    echo "Произошла ошибка при выполнении программы."
          ;;
esac
```

Рис. 3.3: Исходный код скрипта №2

```
[dvshilonosov@dvshilonosov script2]$ bash script.sh
Введите число:
-3
Введите число: -3
Число меньше нуля.
Вы ввели число, меньшее нуля.
[dvshilonosov@dvshilonosov script2]$
```

Рис. 3.4: Результат выполнения скрипта №2

### 3.3 Задание №3

3. Данный скрипт - это командный файл для создания и удаления последовательности файлов с пронумерованными именами от 1 до N. Количество файлов, которые нужно создать, передается в качестве аргумента командной строки. Скрипт проверяет, что передан только один аргумент, затем создает файлы в цикле и выводит сообщение о количестве созданных файлов. После этого скрипт ждет от пользователя ввода любой клавиши и затем удаляет созданные файлы в цикле, снова выводя сообщение о количестве удаленных файлов. (рис. [3.5], [3.6])

```
dvshilonosov@dvshilonosov:~/scripts/script3 — vim script.sh
#!/bin/bash

# Проверяем, что передан один аргумент
if [ $# -ne 1 ]; then
    echo "Usage: $0 <number of files to create>"
    exit 1
fi

# Создаем файлы
for ((i=1; i<=$1; i++)); do
    touch "$i.tmp"
done

echo "Created $1 files."

# Удаляем файлы
read -p "Press any key to delete created files..." -n 1 -r
echo
for ((i=1; i<=$1; i++)); do
    rm -f "$i.tmp"
done

echo "Deleted $1 files."
```

Рис. 3.5: Исходный код скрипта №3

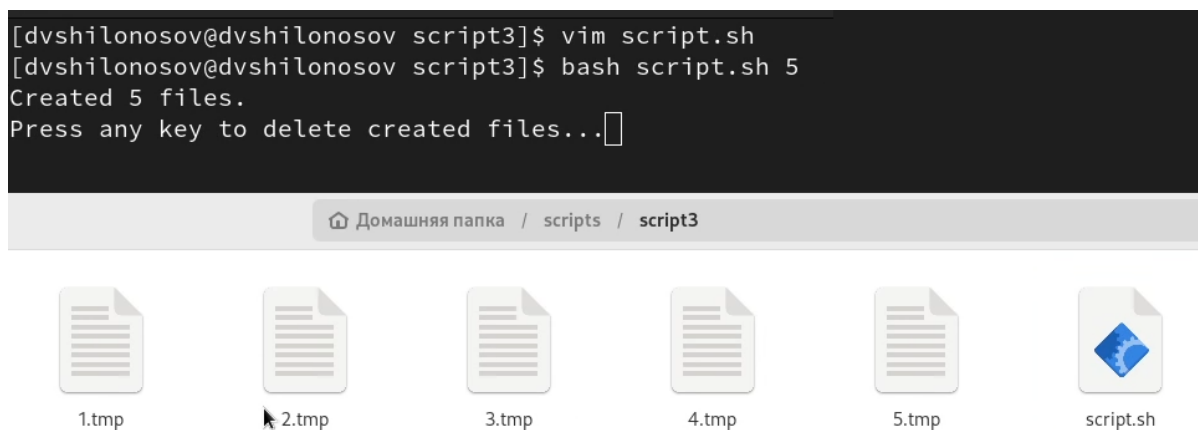
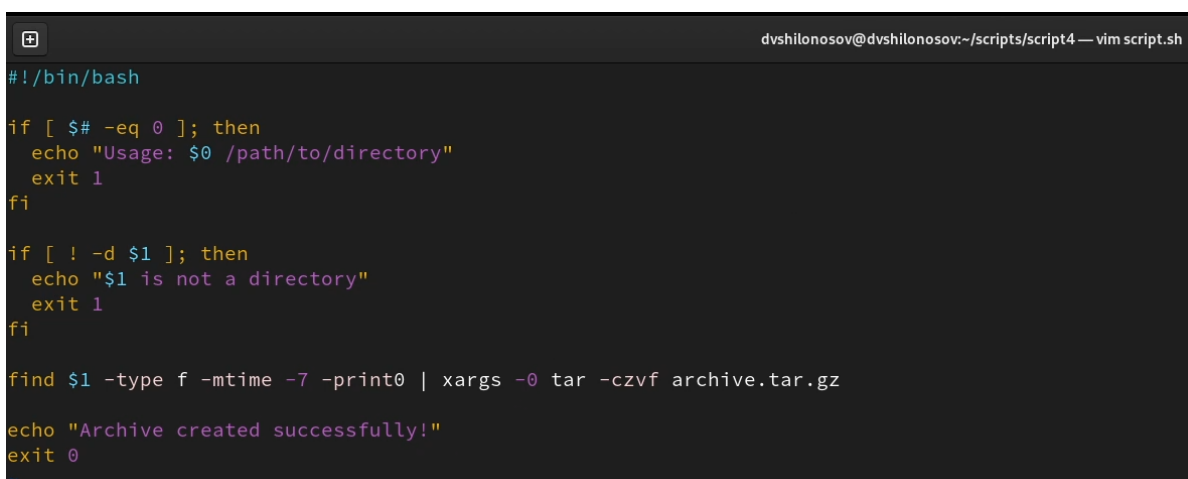


Рис. 3.6: Результат выполнения скрипта №3

### 3.4 Задание №4

4. Этот скрипт - это командный файл для создания архива из файлов, находящихся в указанной директории, которые были изменены менее недели тому назад. Он принимает путь к директории в качестве аргумента при запуске и использует команду `find` для поиска файлов, отвечающих указанному условию. Затем найденные файлы передаются в `tar`, который создает сжатый архив. Если при запуске скрипта не был передан путь к директории или указанный путь не является действительным каталогом, скрипт выдает сообщение об ошибке и завершает работу с кодом ошибки 1. По завершении работы, скрипт выводит сообщение об успешном завершении и устанавливает код возврата 0. (рис. [3.7], [3.8])



```
#!/bin/bash

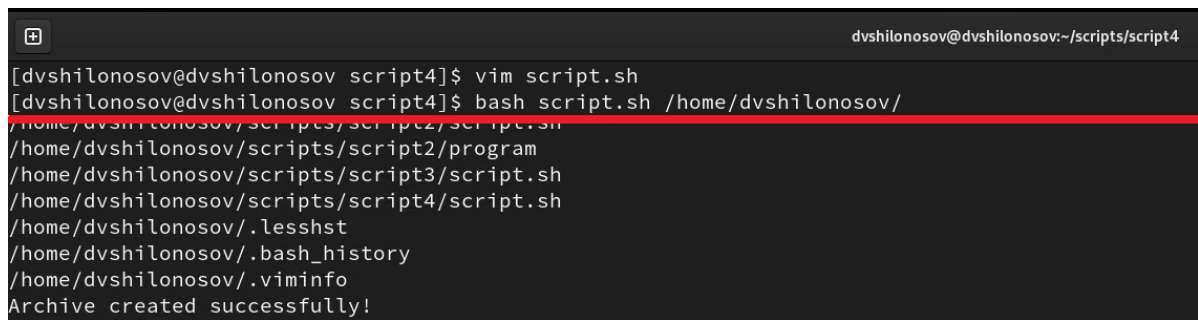
if [ $# -eq 0 ]; then
    echo "Usage: $0 /path/to/directory"
    exit 1
fi

if [ ! -d $1 ]; then
    echo "$1 is not a directory"
    exit 1
fi

find $1 -type f -mtime -7 -print0 | xargs -0 tar -czvf archive.tar.gz

echo "Archive created successfully!"
exit 0
```

Рис. 3.7: Исходный код скрипта №4

A terminal window with a dark background. The title bar at the top right shows the user 'dvshilonosov' and the current directory '~/scripts/script4'. The terminal content shows a sequence of commands and their outputs. The first command is 'vim script.sh'. The second command is 'bash script.sh /home/dvshilonosov/'. The output of this command is a list of files and directories: '/home/dvshilonosov/scripts/script2/script.sh', '/home/dvshilonosov/scripts/script2/program', '/home/dvshilonosov/scripts/script3/script.sh', '/home/dvshilonosov/scripts/script4/script.sh', '/home/dvshilonosov/.lessht', '/home/dvshilonosov/.bash\_history', and '/home/dvshilonosov/.viminfo'. The final line of output is 'Archive created successfully!'.

```
dvshilonosov@dvshilonosov:~/scripts/script4
[dvshilonosov@dvshilonosov script4]$ vim script.sh
[dvshilonosov@dvshilonosov script4]$ bash script.sh /home/dvshilonosov/
/home/dvshilonosov/scripts/script2/script.sh
/home/dvshilonosov/scripts/script2/program
/home/dvshilonosov/scripts/script3/script.sh
/home/dvshilonosov/scripts/script4/script.sh
/home/dvshilonosov/.lessht
/home/dvshilonosov/.bash_history
/home/dvshilonosov/.viminfo
Archive created successfully!
```

Рис. 3.8: Результат выполнения скрипта №4

## 4 Выводы

Были получены навыки написания более сложных командных файлов с использованием логических управляющих конструкций и циклов.