МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Южно-Уральский государственный университет»

(национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

название

ОТЧЕТ

по практической работе №

по дисциплине «Операционные системы семейства Unix/Linux»

Выполнил:

студент группы КЭ–301

/ Д.А. Иванов /

(подпись)

« » 2024 г.

Проверил:

преподаватель кафедры СП

/ П.Д. Ческидов /

(подпись)

« » 2024 г.

Оглавление

[1. Первый bASH-СЦЕНАРИЙ 3](#_Toc160272181)

[2. скрипт для вывода содержимого текущей директории 4](#_Toc160272182)

[3. Сохранение рузультата команд в переменные 5](#_Toc160272183)

[4. Аргумены командной строки 6](#_Toc160272184)

[5. Арифметические операции в скриптах 7](#_Toc160272185)

[6. Скрипт для проверки DNS 8](#_Toc160272186)

[7. Арифметические условные выражения 9](#_Toc160272187)

[8. Строковые условные выражений 10](#_Toc160272188)

[9. Оператор выбора case-esac 11](#_Toc160272189)

[10. Циклический оператор FOR 12](#_Toc160272190)

[11. Циклический оператор while 13](#_Toc160272191)

[12. Функции 14](#_Toc160272192)

# Первый bASH-СЦЕНАРИЙ

**Задача**

Написать и выполнить bash-скрипт, пишущий в стандартный поток вывода сообщения, использую команду **echo**.

Ход работы

Выполнить скрипт возможно выполнить несколькими способами. Во-первых, можно написать название файла, добавив в начало ссылку на текущую директорию, например, «**./myscript.sh**». В этом случае у пользователя должны быть права на выполнение и чтение сценария. Если не добавить ссылку на текущую директорию «**.**», интерпретатор Bash будет искать введенную команду в путях, указанных в переменной окружения **PATH**.

Второй подход заключается в том, чтобы явно передать скрипт интерпретатору Bash на выполнение, указав путь к скрипту в качестве аргумента команде **bash**.

Первая строчка в скриптах имеет особое значение. В неё можно поместить директивы интерпретатора, например, «**#!**» (**shebang**), где указывается путь к интерпретатору, который должен выполнить скрипт.

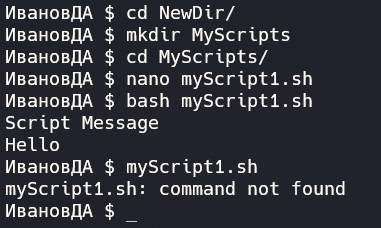


Рисунок 1 – Результат выполнения задания 1

# скрипт для вывода содержимого директории

**Задача**

Написать и выполнить bash-скрипт, выводящий содержимое текущей директории.

Ход работы

Напишем простой скрипт, запускающий команду **ls –l**. В выводе скрипт можно заметить, что у пользователя нет прав на выполнения созданных раннее скриптов.

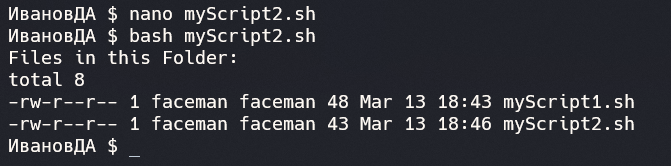


Рисунок 3 – Результат выполнения задания 2

Добавим права на выполнение командой **chmod** с параметрами **a+x**, что означает дать всем пользователям право на исполнениеи запустим скрипт снова.

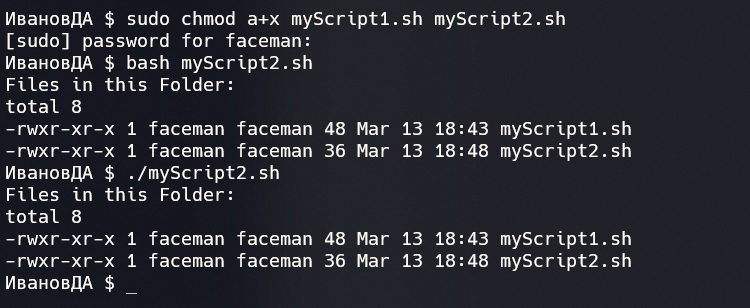


Рисунок 3 – Результат выполнения задания 2

# Сохранение рузультата команд в переменные

**Задача**

Написать и выполнить bash-скрипт, который записывает в переменную **myOS** вывод команды **uname –a**, а затем печатает содержимое переменной в стандартный поток вывода командой **echo**.

Ход работы

Для того, чтобы сохранить вывод команды в переменную необходимо применить механизм подстановки команд (**command substitution**), а затем присвоить результат переменной, воспользовавшись оператором присваивания «**=**». Важно, чтобы между переменной, оператором присваивания и командой не было пустых символов. Для получения содержимого переменной, необходимо к её названию добавить символ доллара «**$**».

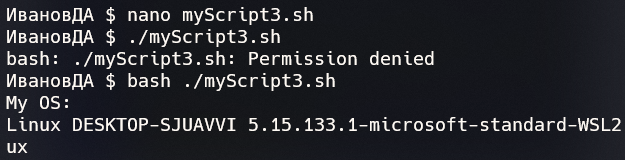


Рисунок 4 – Результат выполнения задания 3

# Аргумены командной строки

**Задача**

Написать и выполнить bash-скрипт, который считывает и выводит аргументы командной строки.

Ход работы

Некоторые переменные в Bash имеют особый смысл. Так, например, переменные с числом от 0 до 9 содержат аргументы командной строки, где нулевой аргумент (**$0**) хранит название запущенного скрипта.

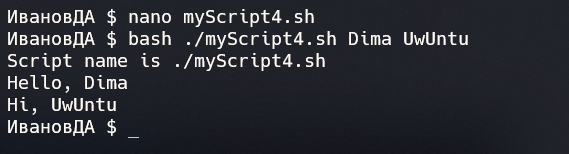


Рисунок 5 – Результат выполнения задания 4

# Арифметические операции в скриптах

**Задача**

Написать и выполнить bash-скрипт, который выводит сумму трех чисел.

Ход работы

Интерпретатор Bash может выполнять простые арифметические и логические операции. Этот механизм получил название арифметическое расширение (arithmetic expansion). Формат арифметического расширения следующий **$(( *expression* ))**.

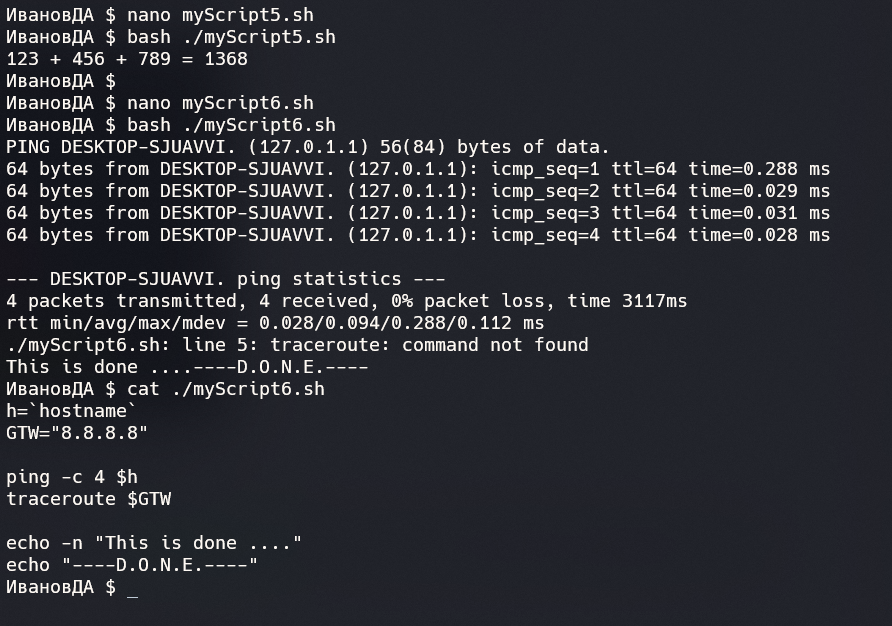


Рисунок 6 – Результат выполнения задания 5

# Скрипт для проверки DNS

**Задача**

Написать и выполнить bash-скрипт для проверки DNS.

Ход работы

Проверить сетевой доступ до собственного хоста можно командой **ping.** Чтобы вывести трассировку до DNS сервера с адресом **8.8.8.8** можно воспользоваться командой **traceroute**.

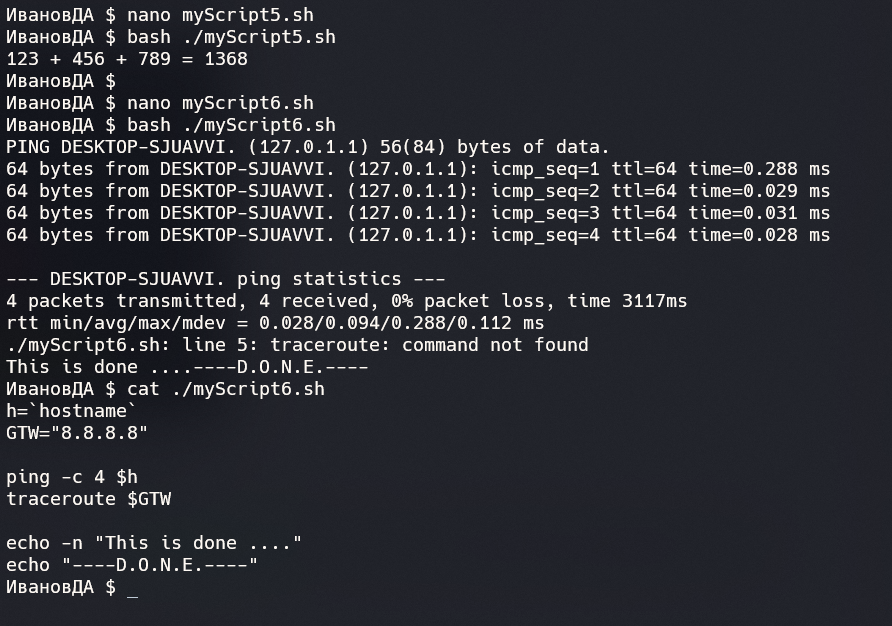


Рисунок 7 – Результат выполнения задания 6

# Арифметические условные выражения

**Задача**

Написать и выполнить bash-скрипт, который получает число из аргументов командой строки и проверяет входит ли оно в диапазон от 5 до 10.

Ход работы

Для управления потоком выполнения используются операторы ветвления **if-elif-else-fi**, которые в зависимости от переданного логического значения выполняют соответствующую ветку кода.

Для сравнения чисел используют **-eq** (равно), **-ne** (не равно), **-gt** (больше чем), **-ge** (больше или равно), **-lt** (меньше чем) и **-le** (меньше или равно) аргументы.

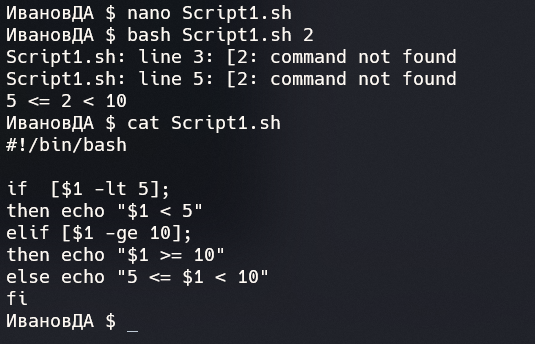


Рисунок 8 – Результат выполнения задания 7

# Строковые условные выражений

**Задача**

Написать и выполнить bash-скрипт, который получает две строки, сравнивает их по лексикографическому порядку и проверяет является ли строки пустыми.

Ход работы

Сравнивать строки по лексикографическому порядку можно используя операторы >, <, = (==). Чтобы проверить, что строка не пустая, можно воспользоваться аргументом **-z**, для проверки на пустоту **-n**.

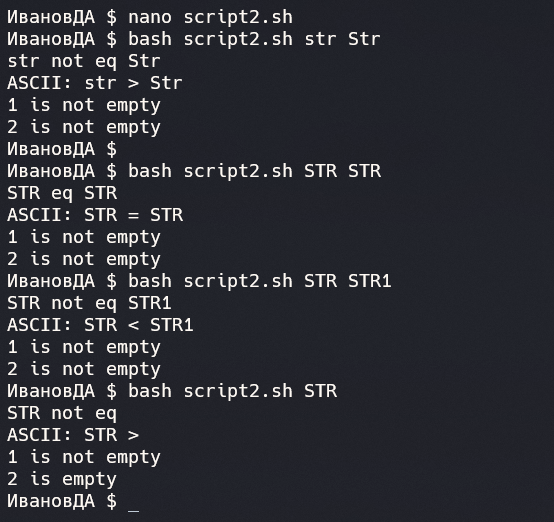


Рисунок 9 – Результат выполнения задания 8

# Оператор выбора case-esac

**Задача**

Написать и выполнить bash-скрипт, который считывает число из стандартного потока ввода и с помощью оператора выбора **case-esac** определяет диапазон, в котором находится это число.

Ход работы

Оператор выбора **case-esac** используется для сопоставления одного значения нескольким условиям.

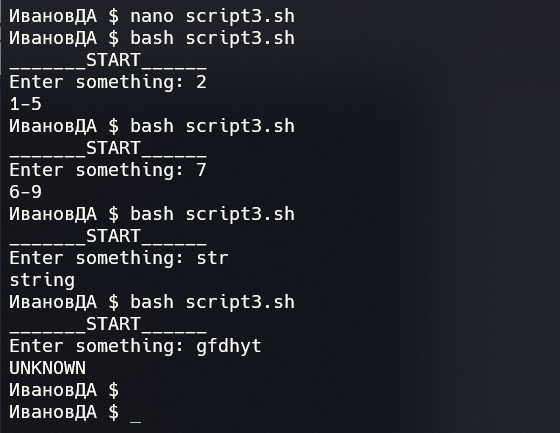


Рисунок 10 – Результат выполнения задания 9

# Циклический оператор FOR

**Задача**

Написать и выполнить bash-скрипт, который итерируется по всем файлам с расширением **txt**, выводит их содержимое. С помощью оператора **for** запишите в файл **myfile.txt** числа от одного до 20 и от 1 до 10 двумя разными способами.

Ход работы

Результат выполнения задания представлен на рисунке 11.

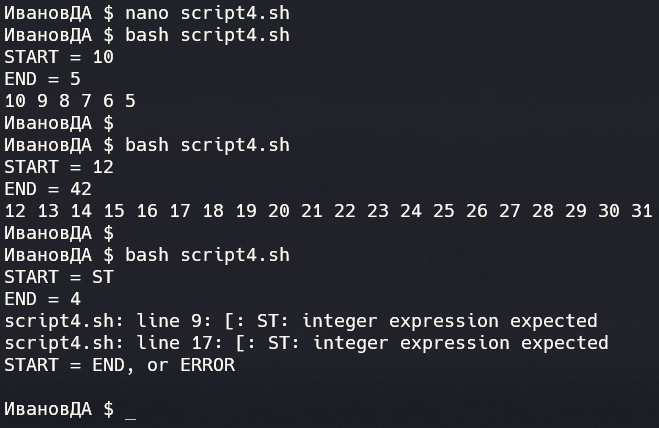


Рисунок 11 – Результат выполнения задания 10

# Циклический оператор while

**Задача**

Написать и выполнить bash-скрипт, в котором используются различные способы итерации с помощью оператора **for**.

Ход работы

Циклический оператор **for** может итерироваться по строке, в этом случае в качестве разделителя выступает пробел. Можно указать другой разделитель, записав его в переменную **IFS**.

На рисунке 12 показан скрипт. Как видно используется 3 способа итерации оператора **for**.

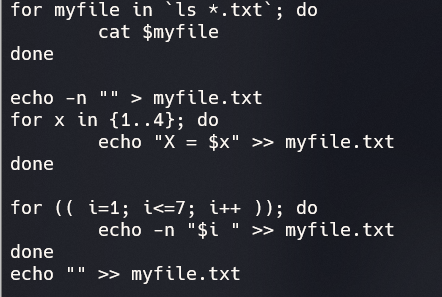


Рисунок 12 – Результат выполнения задания 11

Результат выполнения сценария представлен на рисунке 13.

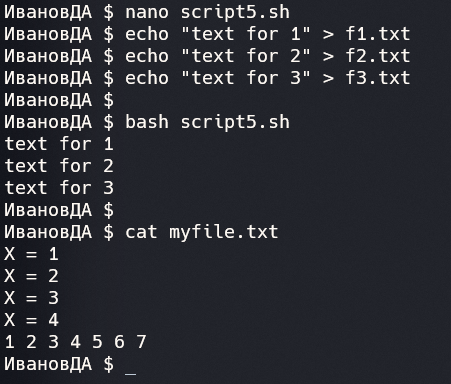


Рисунок 13 – Результат выполнения задания 11

# Функции

**Задача**

Написать функцию в bash-скрипте, которая принимает два числа и вычисляет сумму и произведение этих чисел.

Ход работы

Результат выполнения задания представлен на рисунке 14.

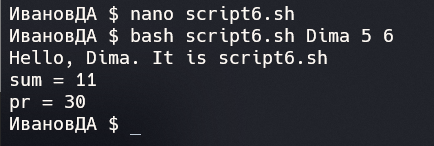


Рисунок 14 – Результат выполнения задания 12