Лабораторная работа №11

Дисциплина: Операционные системы

Гибшер Кирилл Владимирович

Содержание

# 1 Цель работы

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

# 2 Задание

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

а затем ищет в указанном файле нужные строки, определяемые ключом -p

1. 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.
2. 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)
3. 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 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

* С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
* оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
* BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).
* POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

# 4 Выполнение лабораторной работы

1. Используя команды getopts grep, напишем скрипт для первого задания (рис. [[1](#fig:001)])

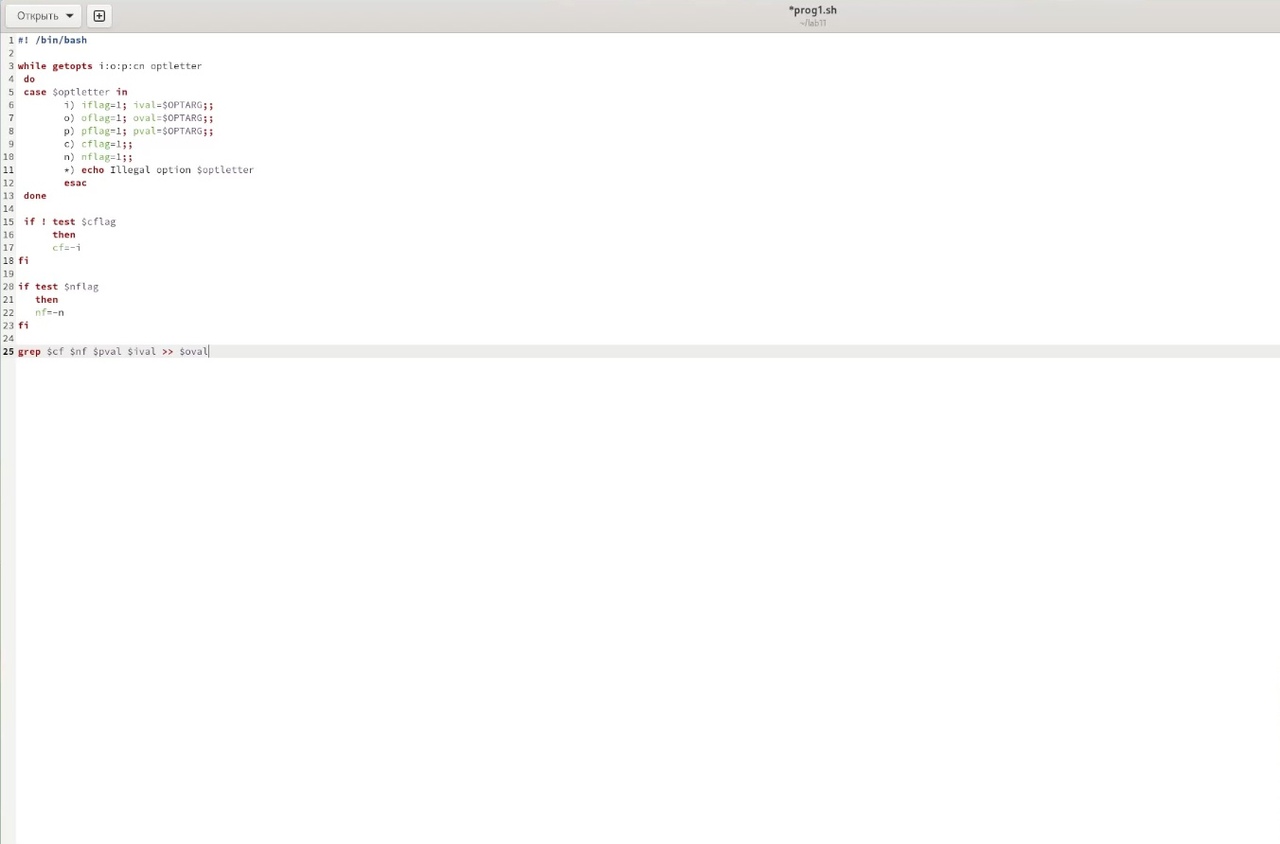


Figure 1: Скрипт первого задания

1. Запустим командный файл и проверим его работоспособность , прописав соответствующие опции в команде. Результат работы проверим в файле output (рис. [[2](#fig:002)])

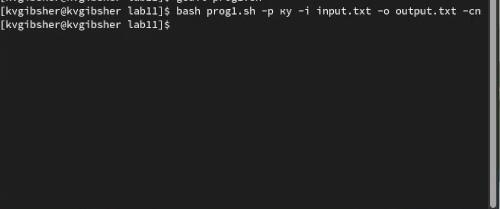


Figure 2: Запуск

1. Содержимое файла input (рис. [[3](#fig:003)])

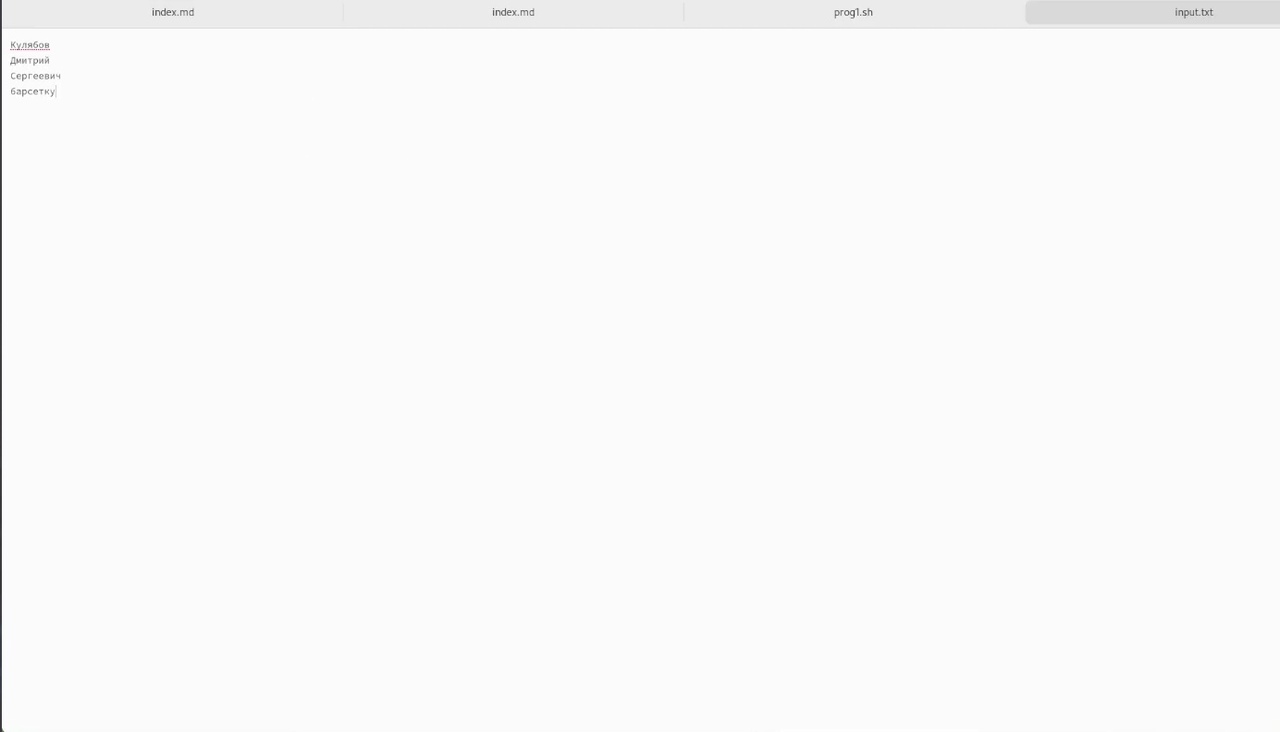


Figure 3: input.txt

1. Содержимое файла output (рис. [[4](#fig:004)])

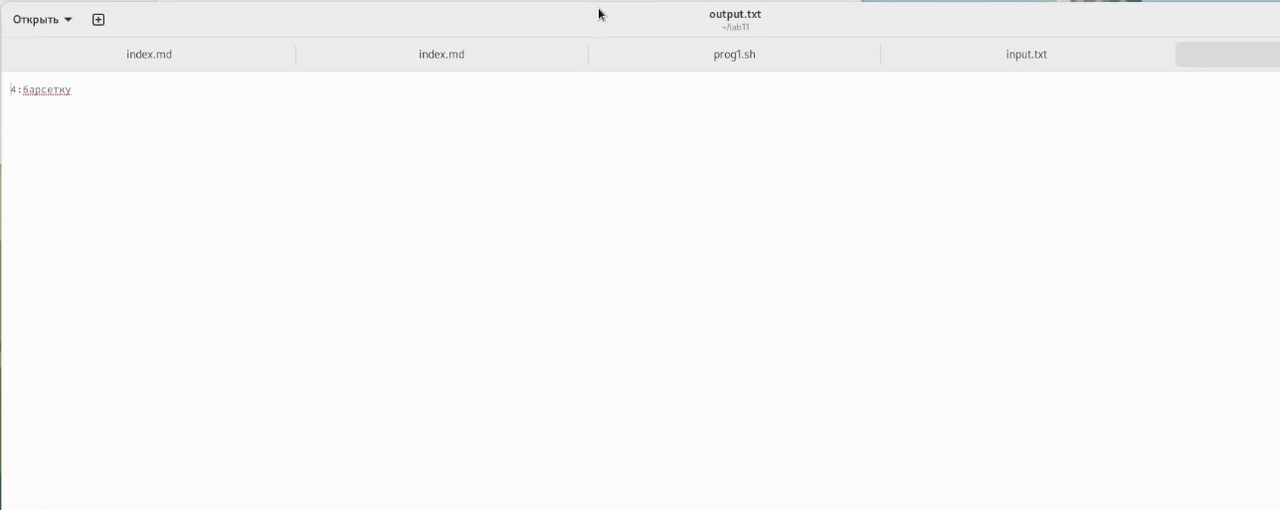


Figure 4: output.txt

1. Напишем код для файла с поддержкой Си (рис. [[5](#fig:005)])

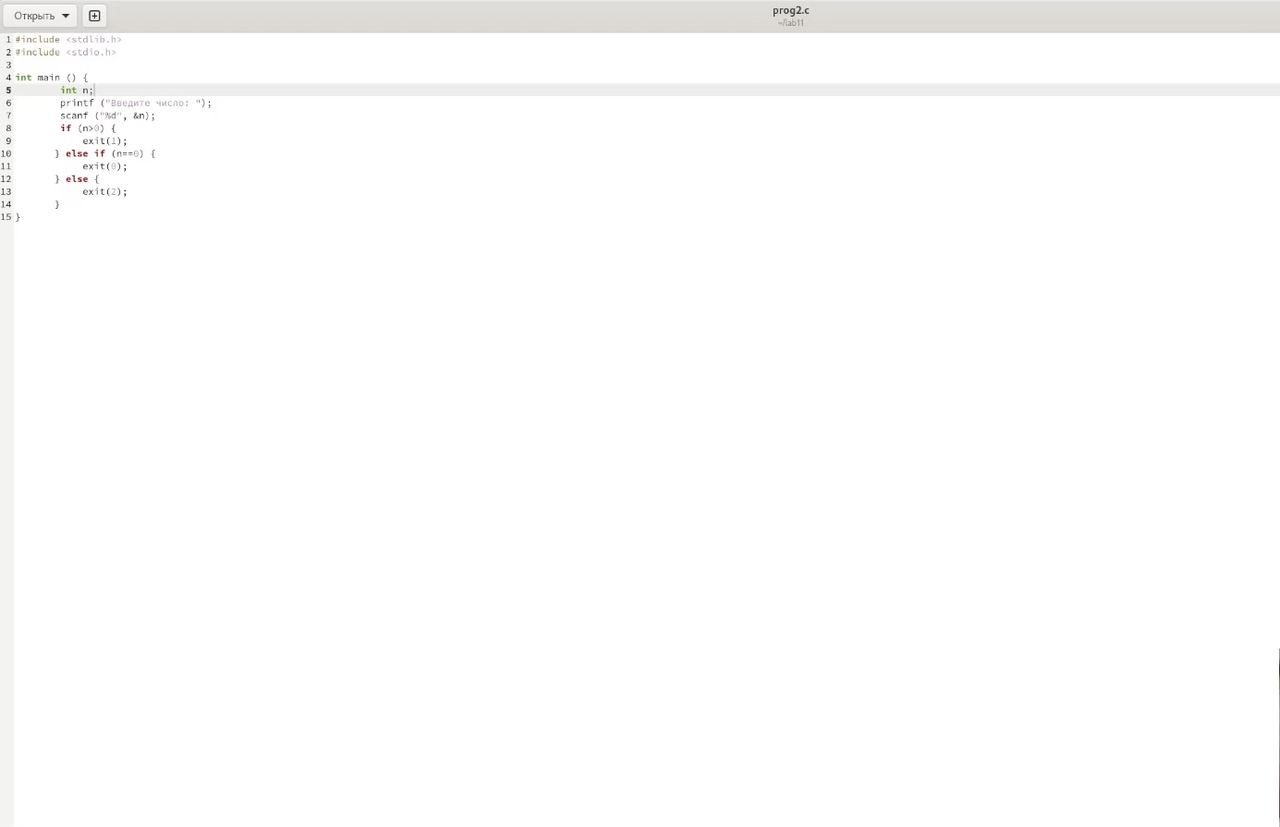


Figure 5: Файл си

1. Используя раннее написанный код СИ , напишем скрипт для командного файла, который будет удовлетворять условия задания 2 (рис. [[6](#fig:006)])

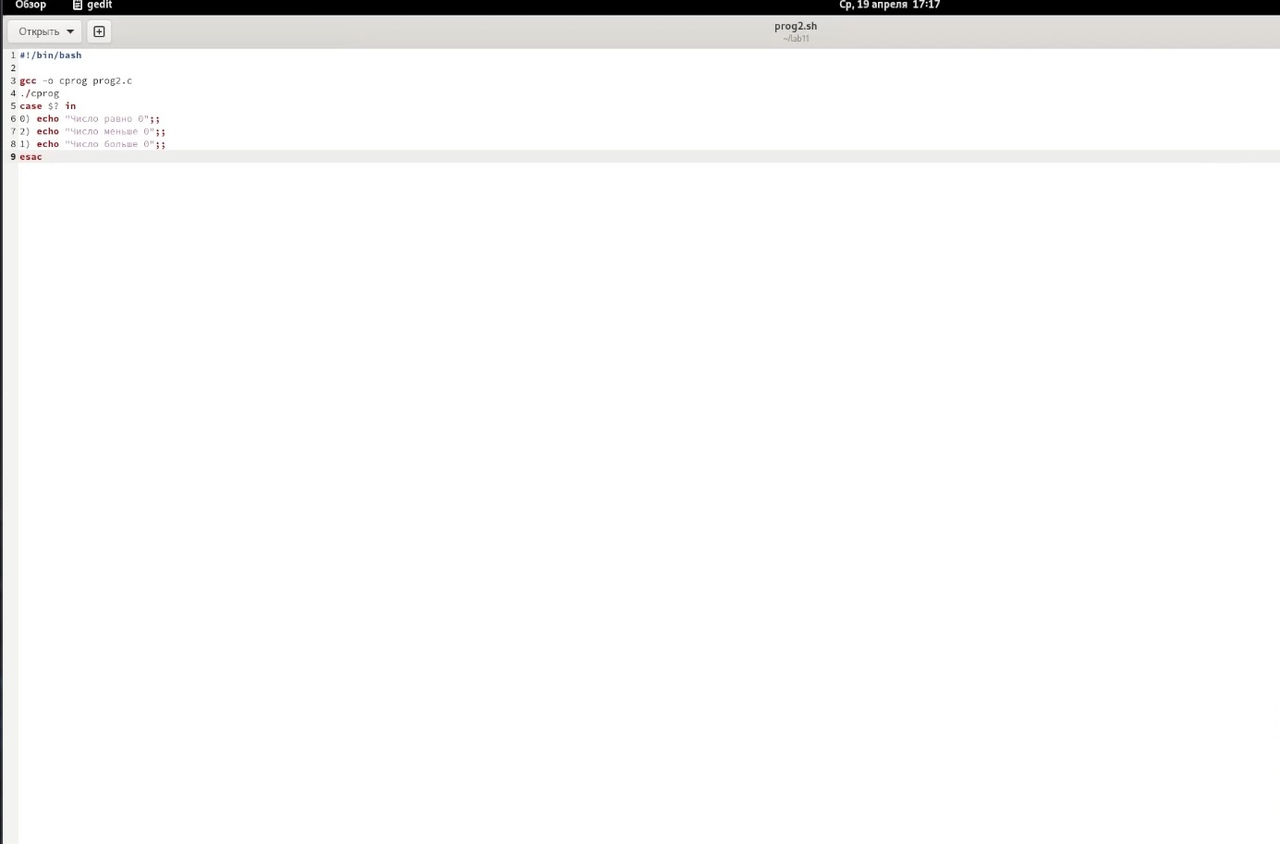


Figure 6: Командный файл

1. Запуск командного файла и проверка работоспособности кода. Действительно, все необходимые проверки над вводимыми пользователем числами проводятся. Задание выполнено (рис. [[7](#fig:007)])

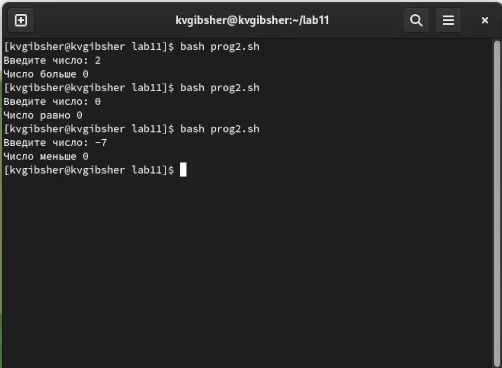


Figure 7: Запуск и проверка работоспособности кода

1. Напишем скрипт для третьего задания (рис. [[8](#fig:008)])

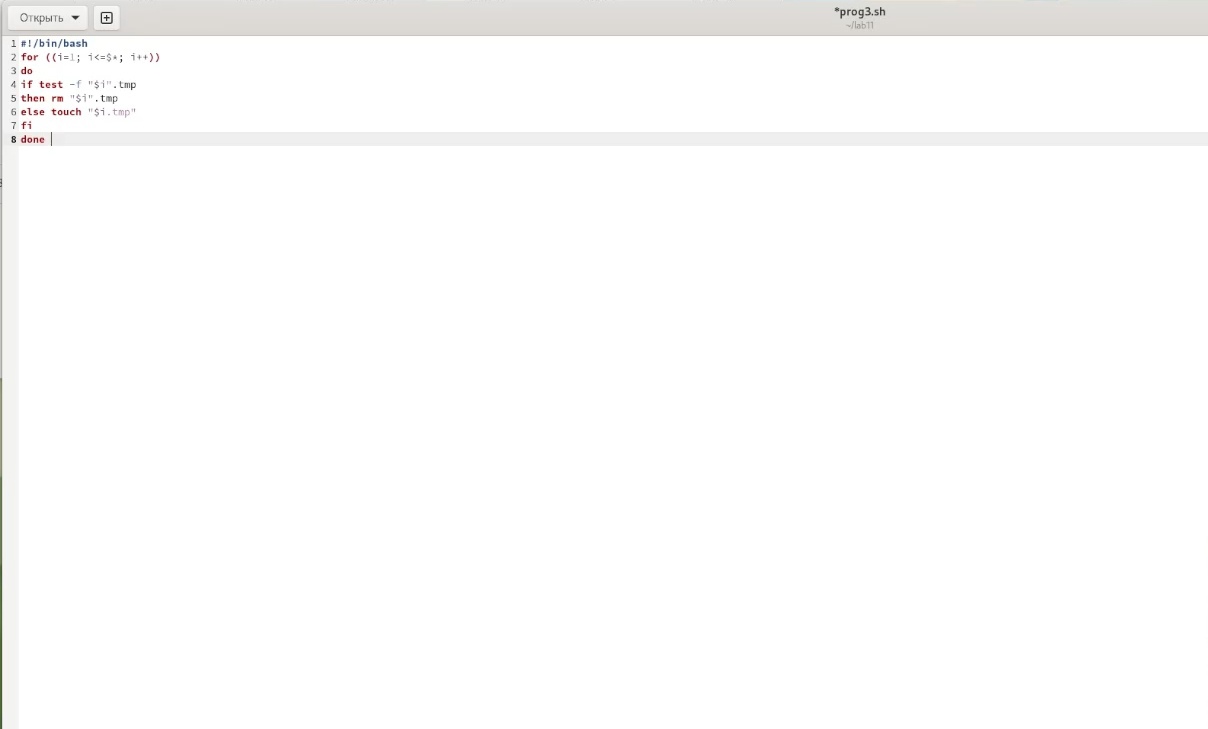


Figure 8: Скрипт третьего задания

1. Запустим командный файл и увидим, что при первом запуске создается 3 файла тмп и при повторном запуске, созданные им файлы удаляются, тем самым программа соответствует условиям задания. (рис. [[9](#fig:009)])

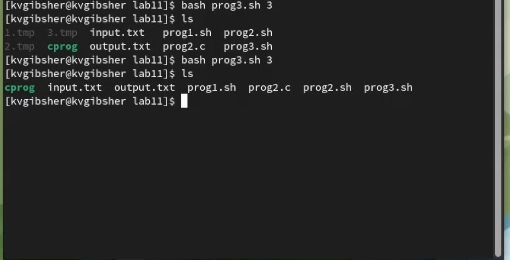


Figure 9: Запуск и проверка

1. Напишем скрипт для последнего задания (рис. [[10](#fig:010)])

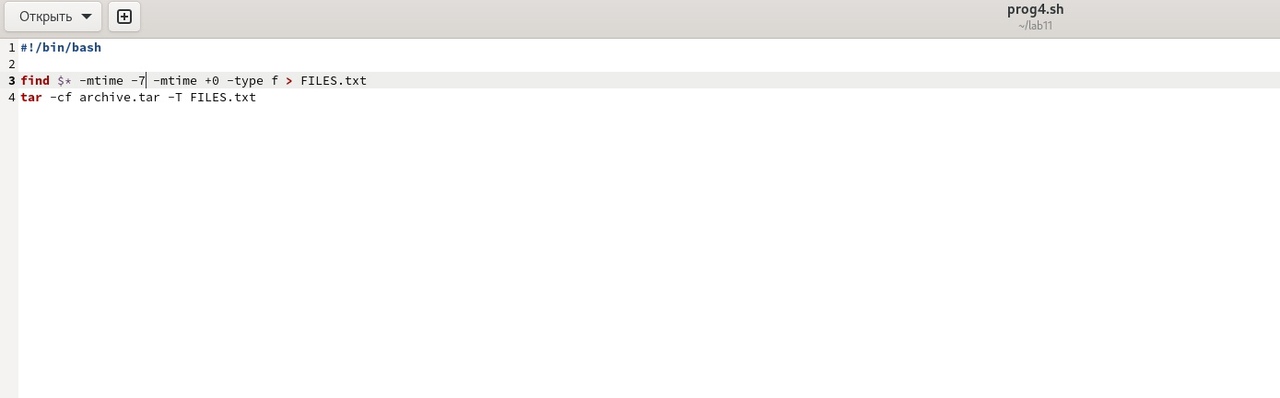


Figure 10: Скрипт последнего задания

1. Запустим командный файл и увидим, что при запуске архив и файл FILES.txt были успешно созданы. (рис. [[11](#fig:011)])

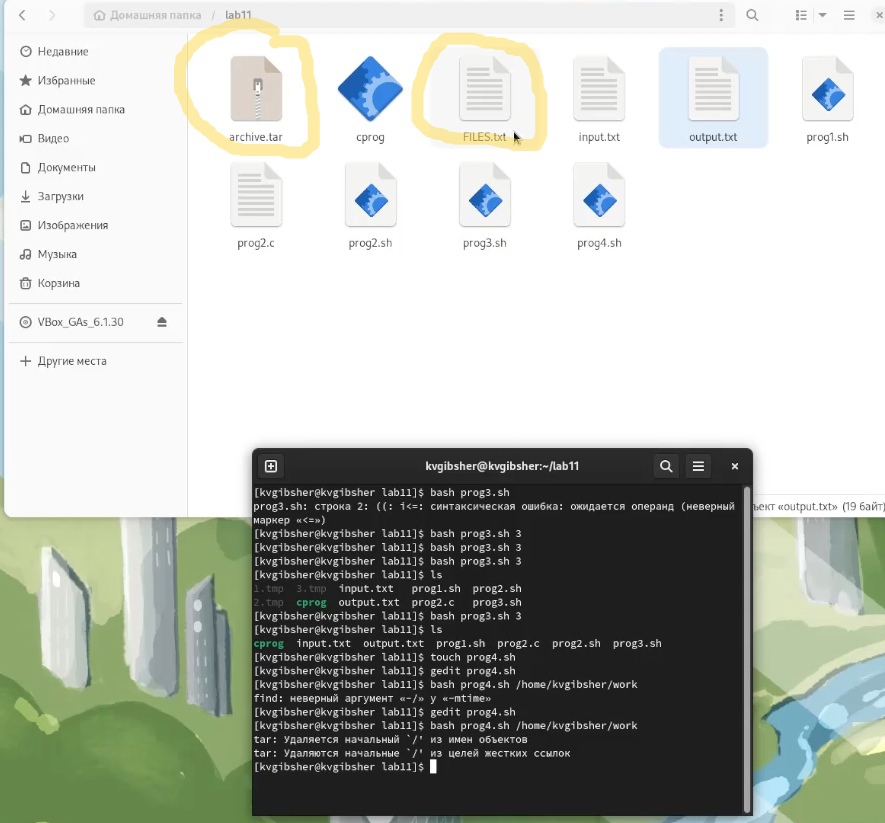


Figure 11: Запуск исполняемого файла

1. Проверим результат, открыв файл FILES.txt , в которой и указываются все архивированные файлы (рис. [[12](#fig:012)])

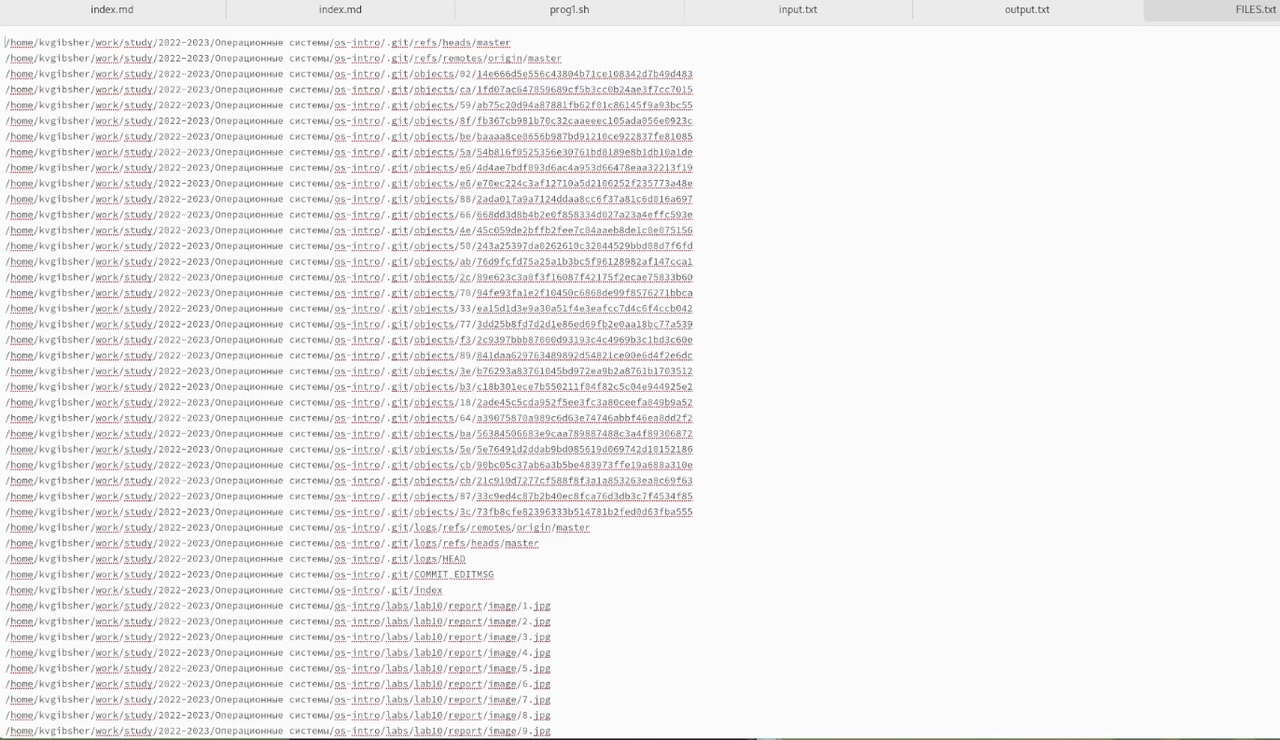


Figure 12: Резульатт

# 5 Вывод

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