Лабораторная работа №13

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Полякова Юлия Александровна

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. Ознакомиться с теоретическим материалом.
2. Написать программы.
3. Ответить на контрольные вопросы.

# 3 Выполнение лабораторной работы

1. Рассмотрим первый скрипт по заданию: 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.(рис. 1).

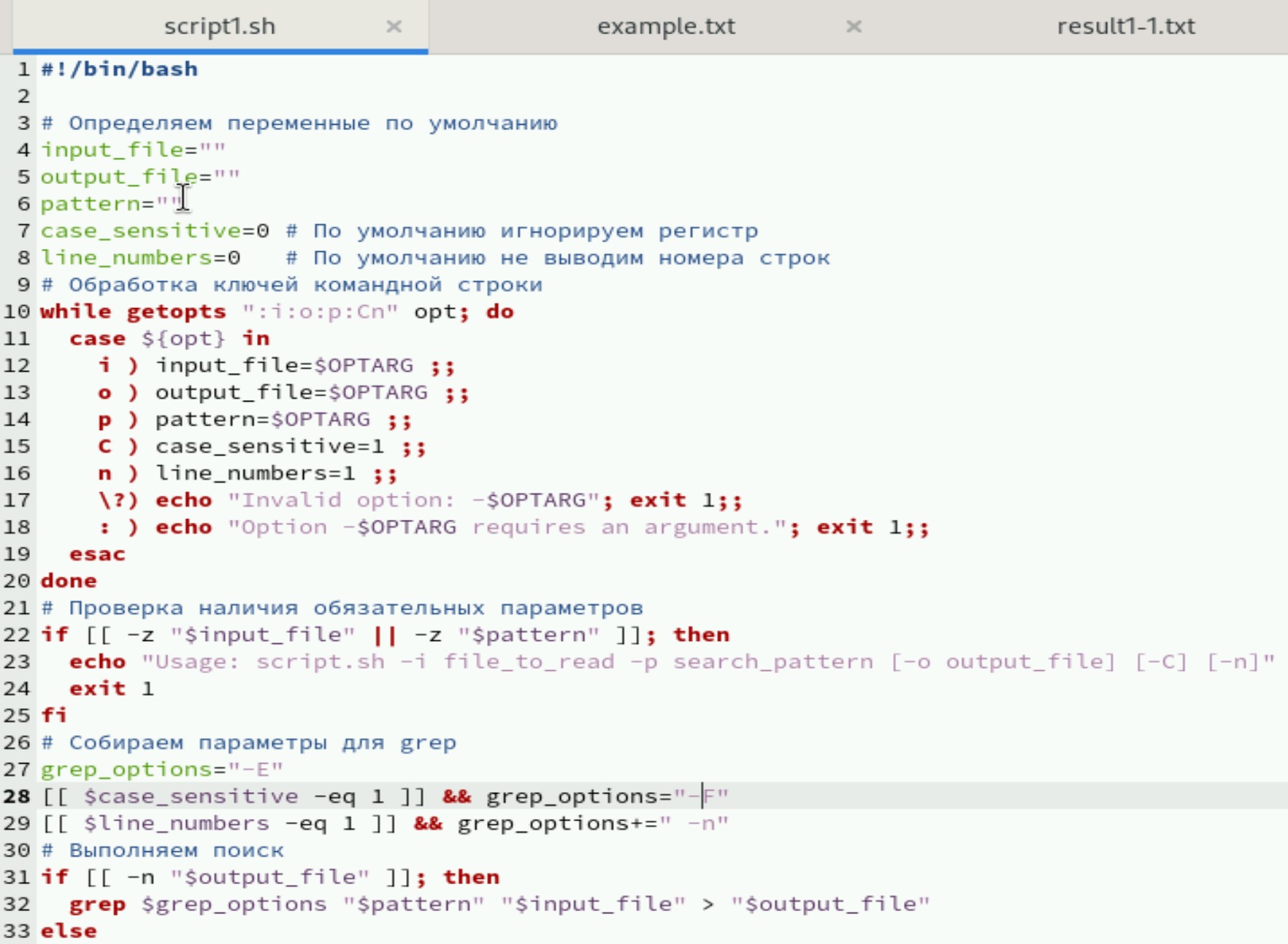


Рис. 1: Листинг script1.sh

1. Открываем доступ к файлу и запускаем. В начале мы запустили поиск “hello” в файле “example.txt” без учета регистра и номеров строк в файл result1-1.txt А затем проверили учет больших букв и номера строк и вывели результат сначала в терминал, а потом в result1-2.txt (рис. 2)

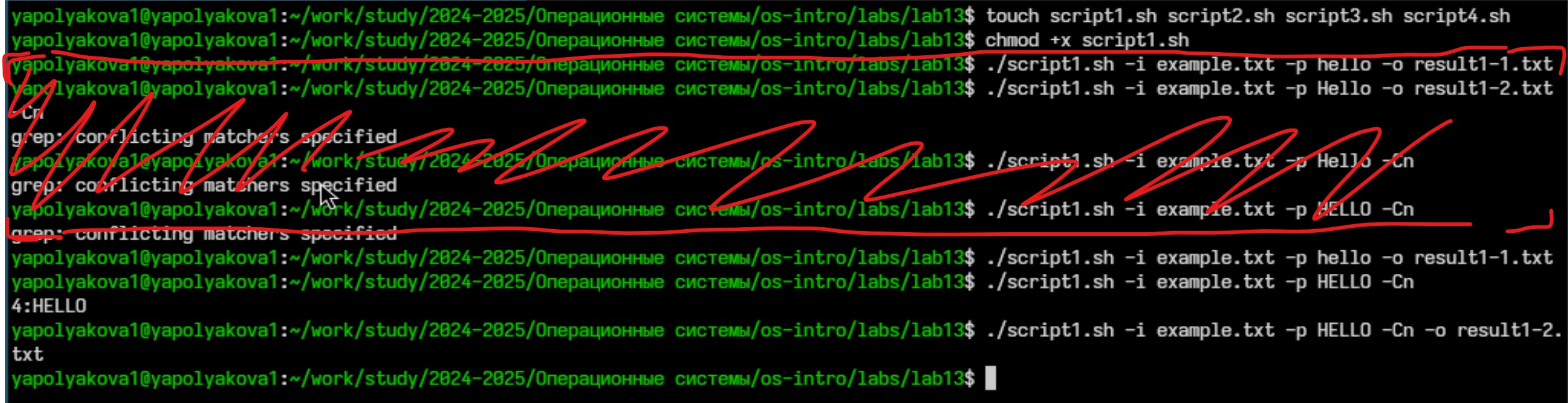


Рис. 2: Запуск script1.sh

1. Файл example.txt (рис. 3)

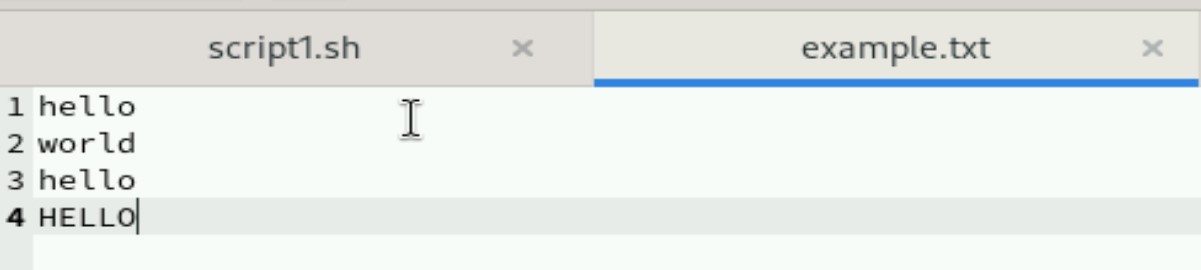


Рис. 3: example.txt

1. Файл result1-1.txt (рис. 4)

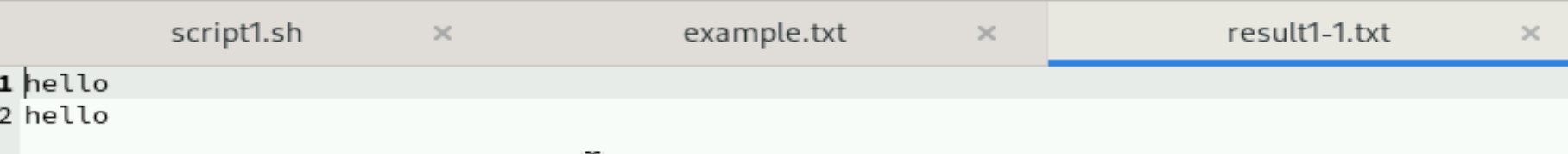


Рис. 4: result1-1.txt

1. Файл result1-2.txt (рис. 5)

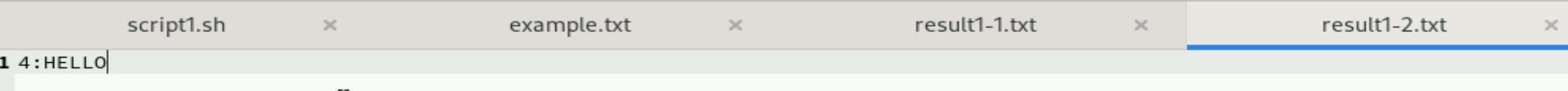


Рис. 5: result1-2.txt

1. Рассмотрим второе задание. Сначала надо было “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.” На фото листинг этой программы (рис. 6)

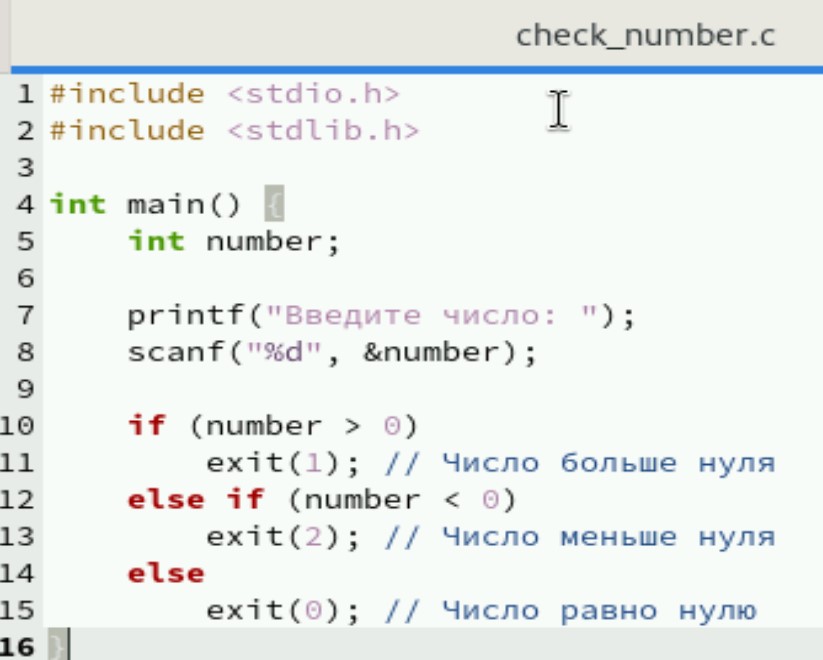


Рис. 6: Листинг check\_number.c

1. Теперь рассмотрим сам командный файл. Задание: Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено. (рис. 7)

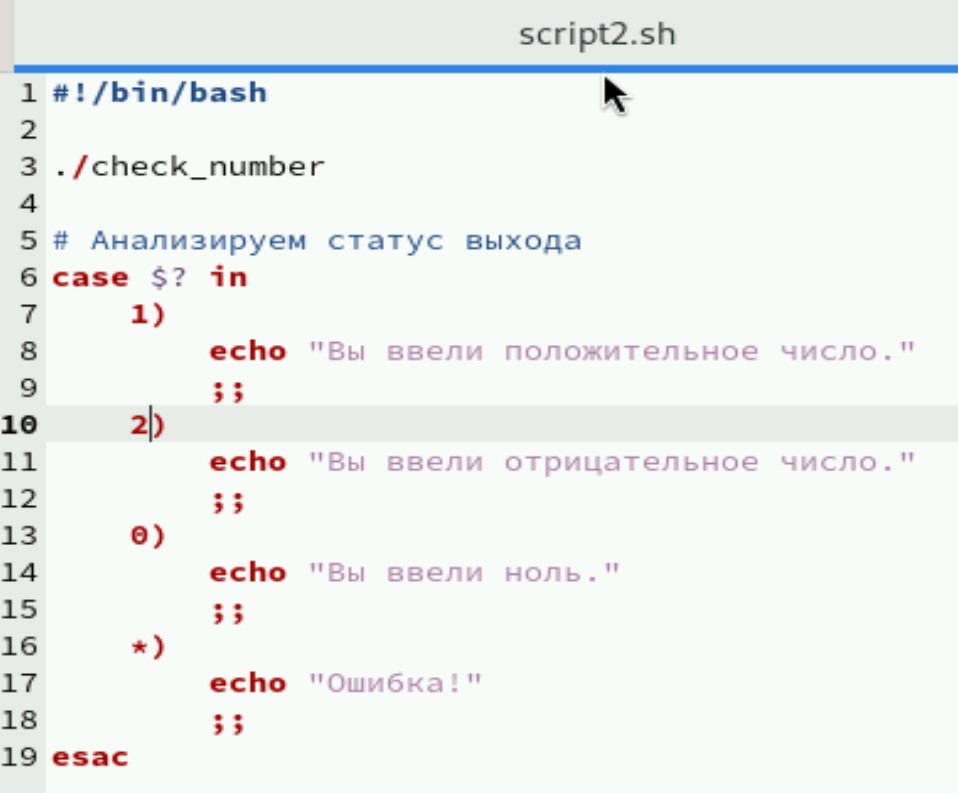


Рис. 7: Листинг script2.sh

1. Компилируем C-файл с помощью gcc, открываем доступ к командному файлу и запускаем. Тестируем с разными числами (рис. 8)

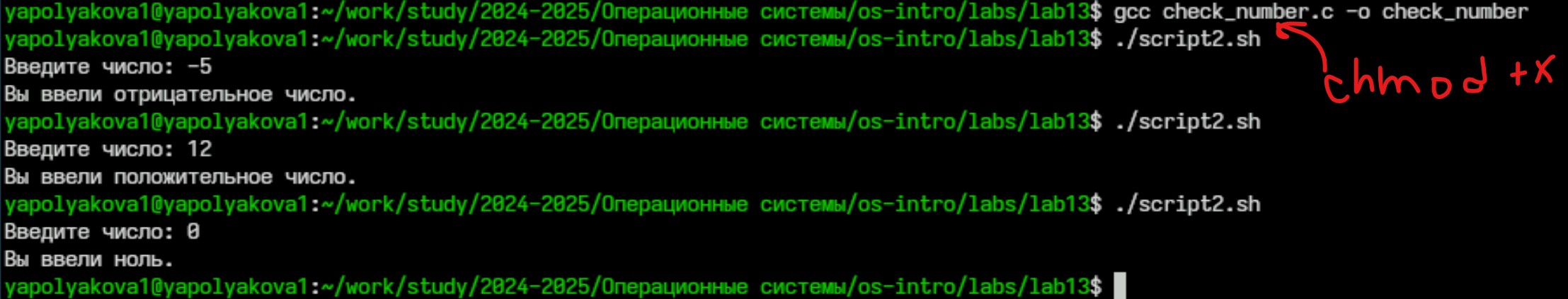


Рис. 8: Запуск script2.sh

1. Рассмотрим третий скрипт. Задание: 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). Я реализовала 2 режима - создание и удаление (рис. 9)

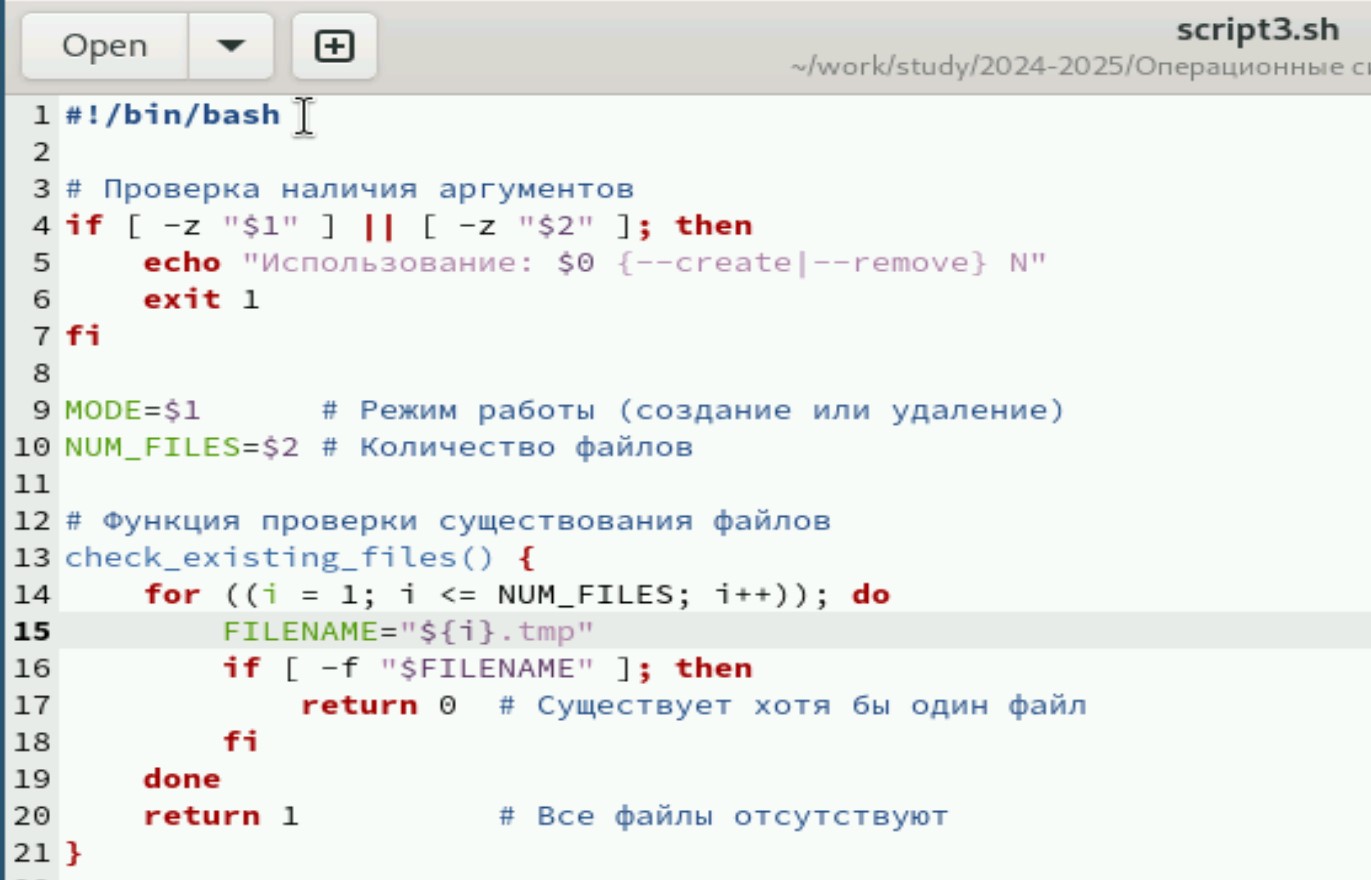


Рис. 9: Листинг script3.sh - часть 1 (проверка аргум. и сущ-я файлов)

Основная часть кода script3.sh, работа с режимами (рис. 10)

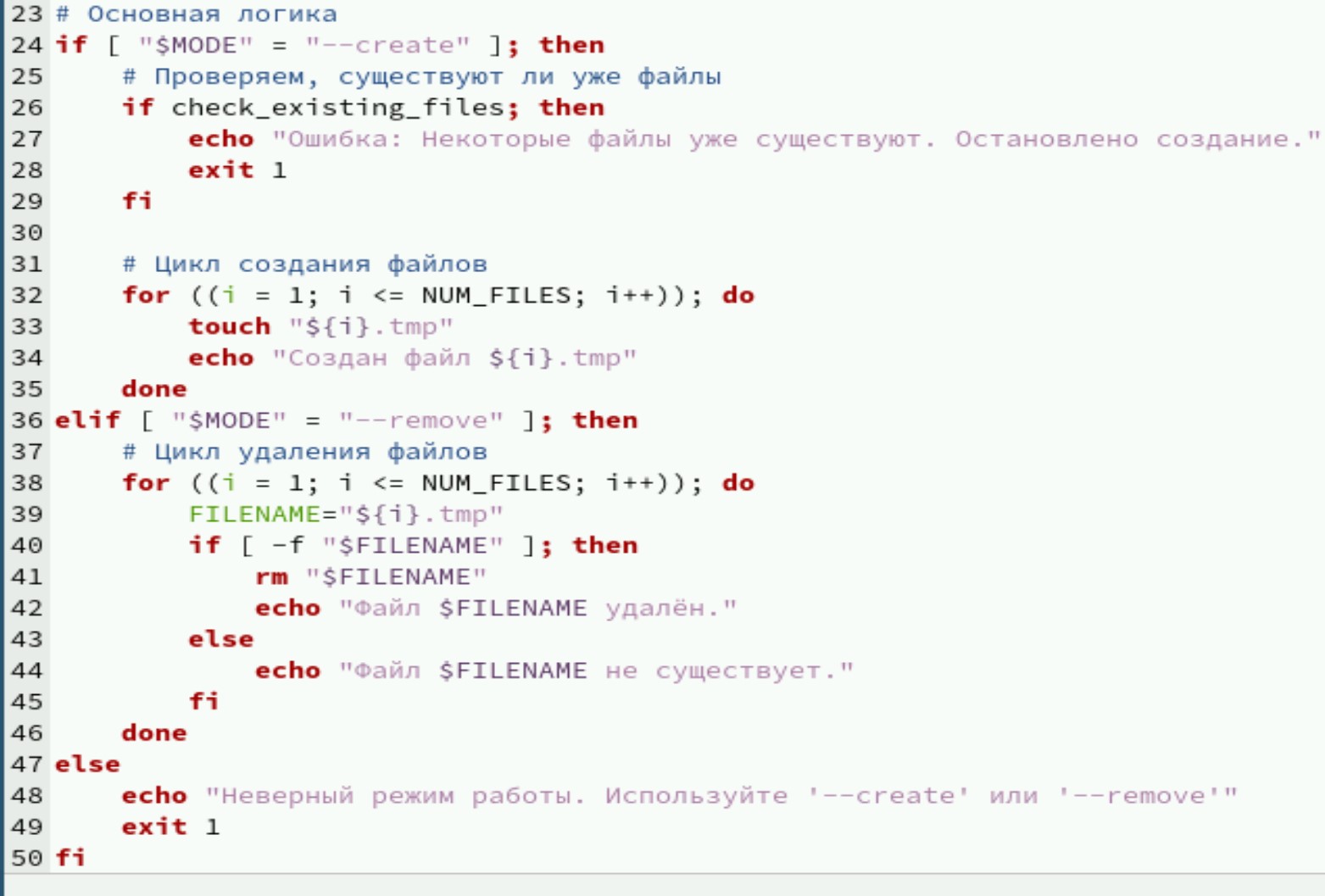


Рис. 10: Листинг script3.sh - часть 2 (основная часть с режимами)

1. Открываем доступ к исполнению, тестируем на создании и удалении 3-х файлов (рис. 11)

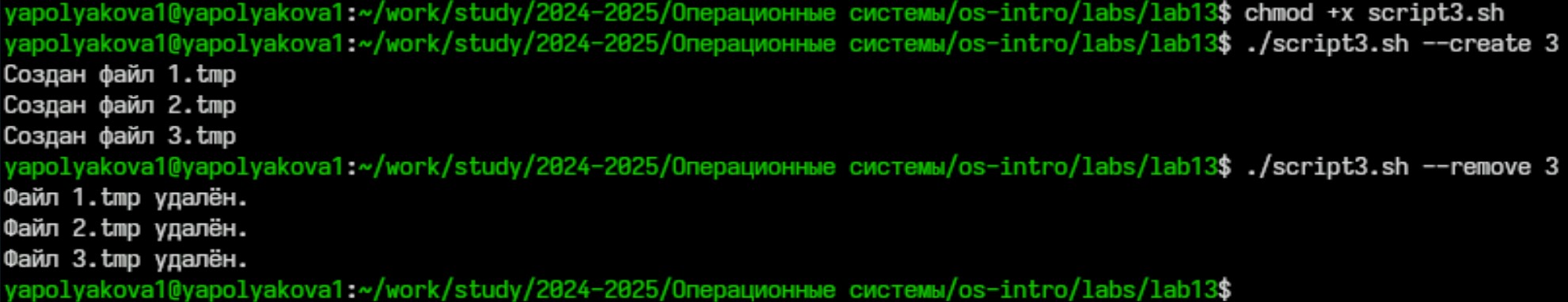


Рис. 11: Запуск script3.sh

1. Рассмотрим 4-ый командный файл. Задание: 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). (рис. 12)

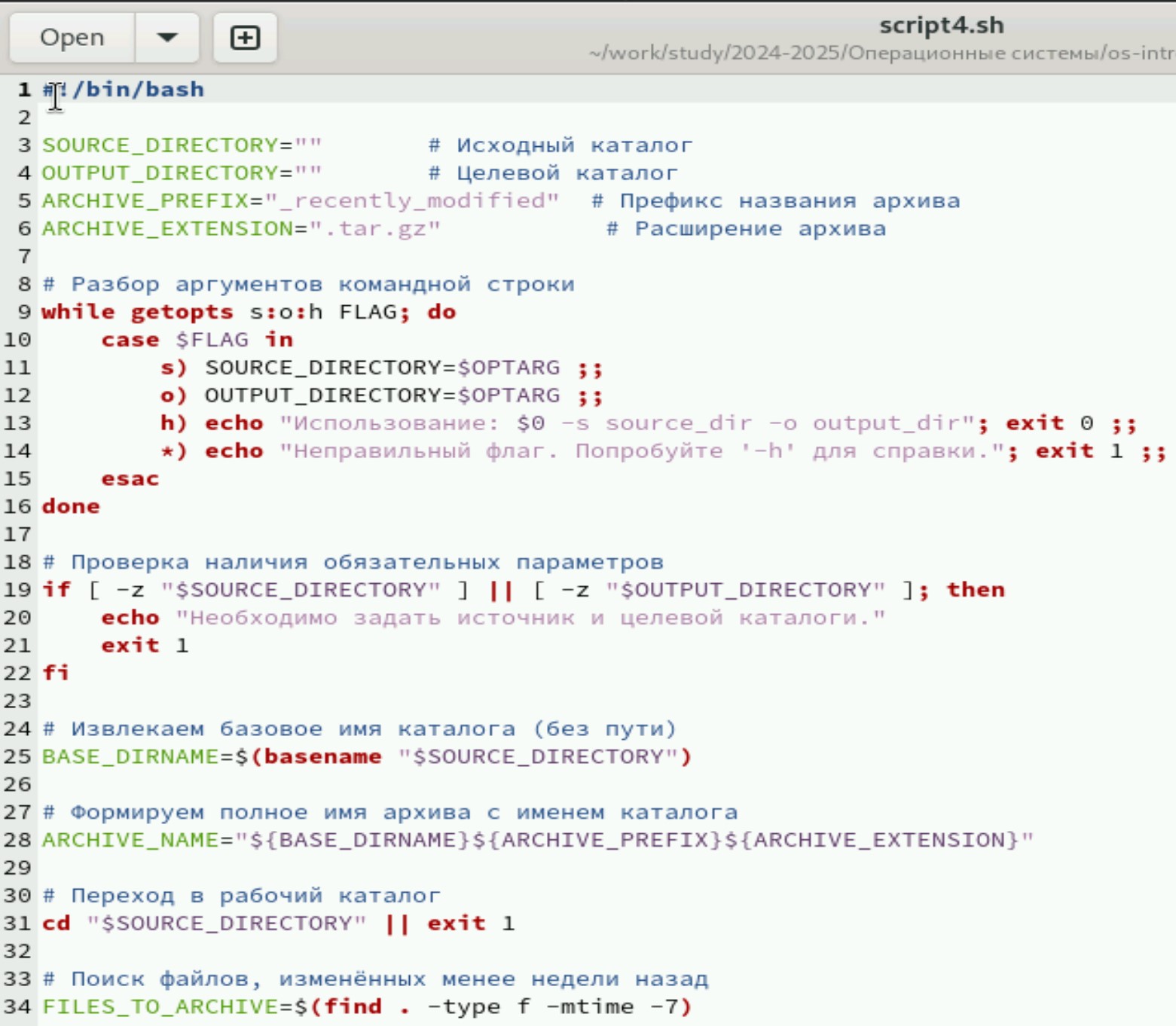


Рис. 12: Листинг script4.sh - часть 1

Конец скрипта с архивацией (рис. 13)



Рис. 13: Листинг script4.sh - часть 2

1. Открываем доступ к исполнению, запускаем, архивируем файлы этой лабораторной работы (рис. 14)

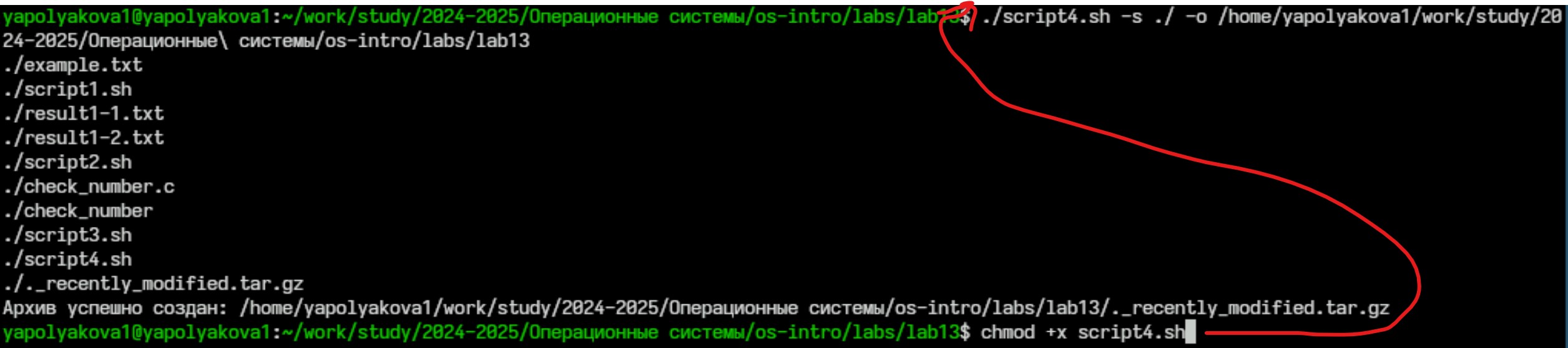


Рис. 14: Запуск script4.sh

1. Архив действительно создался (рис. 15)

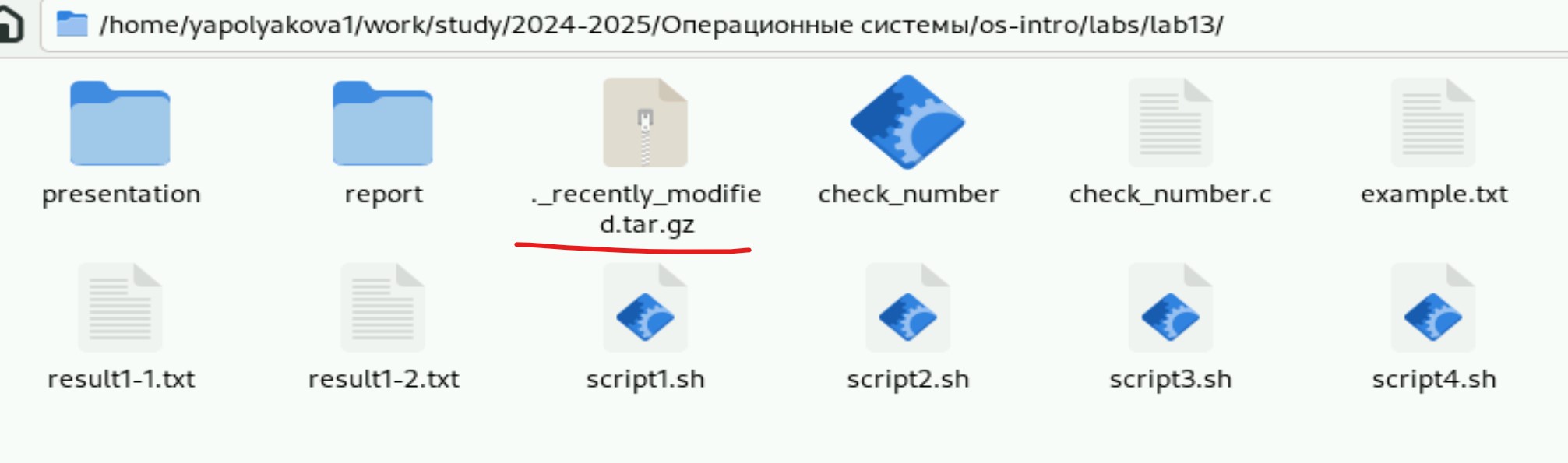


Рис. 15: Архив

# 4 Контрольные вопросы

1. Каково предназначение команды getopts?

Команда getopts предназначена для удобной обработки аргументов командной строки в сценариях оболочки (shell scripts). Она помогает разбивать переданные аргументы на отдельные элементы и проверять правильность передачи флагов и значений. Обычно применяется совместно с циклом while, что позволяет автоматизировать обработку сложных комбинаций параметров, обеспечивая безопасность и ясность синтаксиса.

Пример использования:

while getopts ab:c OPT; do  
 case $OPT in  
 a) echo "Опция A была указана";;  
 b) echo "Опция B имеет значение: $OPTARG";;  
 c) echo "Опция C указана";;  
 ?) echo "Неправильно использован флаг";;  
 esac  
done

1. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы (такие как звездочка \*, знак вопроса ?, квадратные скобки []) играют важную роль в построении масок для поиска и подбора файлов в операционной системе. Они позволяют удобно находить группы файлов по общим признакам.

Примеры:

* Маска \*.txt выберет все файлы с расширением .txt.
* Маска file?.dat подберёт файлы с названием длиной в четыре символа, начинающиеся с file и оканчивающиеся на .dat.

Эти маски часто применяются в командах типа ls, rm, cp, что значительно ускоряет выбор файлов.

1. Какие операторы управления действиями вы знаете?

Операторы управления помогают контролировать поток выполнения инструкций в сценарии оболочки. Основные операторы включают:

* **if** — оператор условного ветвления, позволяющий исполнять блок кода только при выполнении какого-либо условия.
* **case** — оператор множественного выбора, похожий на switch-case в языках программирования.
* **for**, **while**, **until** — операторы организации циклов.
* **break**, **continue** — операторы изменения хода выполнения цикла.
* **&&** (логическое AND) и **||** (логическое OR) — используются для объединения нескольких условий или команд.

1. Какие операторы используются для прерывания цикла?

Для остановки цикла используются два оператора:

* **break** — немедленно выходит из ближайшего охватывающего цикла (for, while, until).
* **continue** — пропускает оставшуюся часть итерации и переходит сразу к следующей итерации цикла.

Пример использования операторов:

for i in $(seq 1 10); do  
 if [ $i -gt 5 ]; then break; fi  
 echo $i  
done

1. Для чего нужны команды false и true?

Командами false и true являются стандартные команды оболочки, используемые для простых тестов условий. Их назначение следующее:

* **false** — всегда возвращает ненулевой код завершения (обычно 1), обозначая ошибку или ложное условие.
* **true** — всегда возвращает нулевой код завершения (0), обозначая успешное завершение или истинное условие.

Они полезны в случаях, когда необходимо организовать простую проверку или задать заведомо известное условие.

Пример использования:

if true; then  
 echo "Это всегда правда"  
fi

1. Что означает строка if test -f mani.$s, встреченная в командном файле?

Эта строка представляет собой проверку существования файла с определенным форматом. Рассмотрим подробнее:

* test -f filename — проверяет, существует ли файл и является ли он обычным файлом (не каталогом, символьной ссылкой и т.п.).
* man$s/$i.$s — выражение, генерирующее динамическое имя файла. Здесь:
  + $s — вероятно, какая-то переменная, заменяющая расширение или категорию файла.
  + $i — возможно, индекс или порядковый номер файла.

Таким образом, данная конструкция проверяет, существует ли конкретный файл с определенной структурой имени.

1. Объясните различия между конструкциями while и until.

Конструкция while: цикл выполняется, пока условие истинно. Иначе говоря, цикл выполняется снова и снова, пока условие возвращает успех (нулевой код завершения).

Конструкция until: цикл выполняется, пока условие ложно. Таким образом, цикл выполняется повторно, пока условие возвращает неудачу (ненулевой код завершения).

# 5 Вывод

Были изучены основы программирования в оболочке ОС UNIX. Мы научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.