МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем

и технологий

**Отчет**

**по лабораторной работе № 3**

по дисциплине: ”Системное программирование”

на тему: ***”*Командный язык и скрипты Shell*”***

Выполнил**:** студент группы *10702121* Писарик А.С

Филанович Д.В.

Принял**:** пр. Давыденко Н. В.

Минск 2023

# Лабораторная работа №3.

**Цель работы:** Закрепить на практике основы работы со сценариями командной оболочки и перенаправление ввода/вывода.

## Задание 1

1. Написать скрипт, который записывает содержимое одного файла в другой и при этом оба файла существуют до операции. Дать два разных варианта выполнения.

2. Найти все файлы, начинающиеся на слово script и записать их имена в файл list.txt.

3. Написать скрипт, который проверяет, содержится ли скриптовый файл в директории, если да, то выполнить его.

4. Написать скрипт, подсчитывающий сумму от 1 до 10.

5. А какие командные интерпретаторы установлены в вашей системе?

Напишите скрипт поиска местонахождения командных интерпретаторов вашей системы. Также определите какой интерпретатор используется сейчас в вашей системе?

Решение

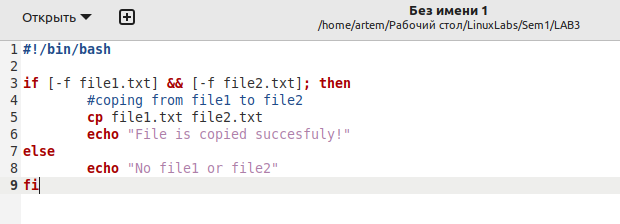
**1.** Написать скрипт, который записывает содержимое одного файла в другой и при этом оба файла существуют до операции. Дать два разных варианта выполнения.

1. Вызываем текстовый редактор

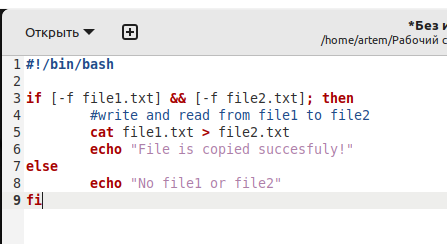


1. Пишем код сценария и сохраняем файл

1) Используя команду `cp` для копирования файла:



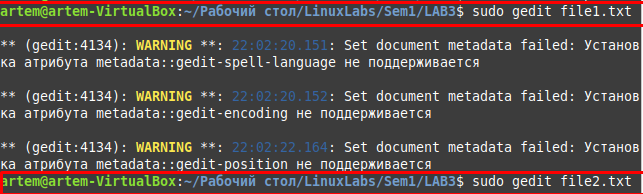
2) Используя оператор `cat` для чтения и записи файлов:

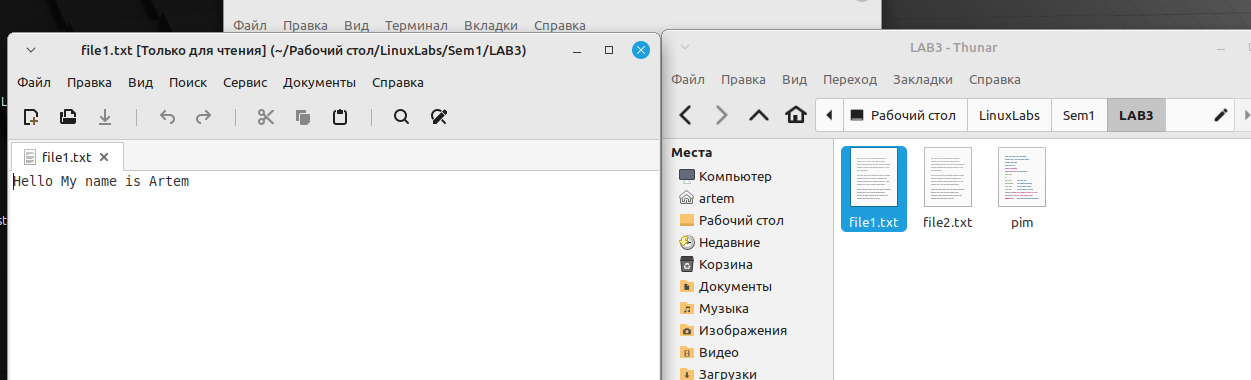


1. Делаем скрипт выполняемым файлом



4. Создаем два текстовых файла и заносим текст в один из них.





5. Запускаем скрипт



6. Проверяем наш второй файл:

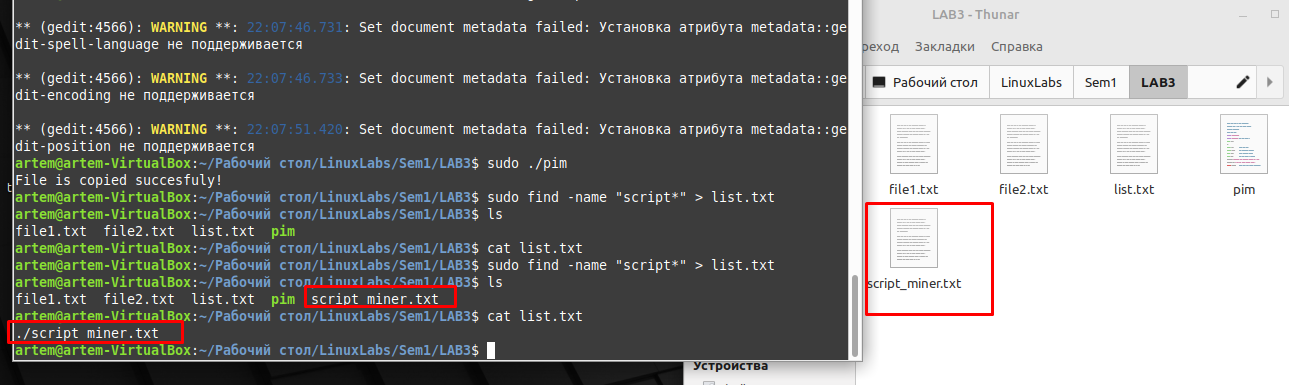


**2.** Найти все файлы, начинающиеся на слово script и записать их имена в файл list.txt.

1. Выполняем команду find

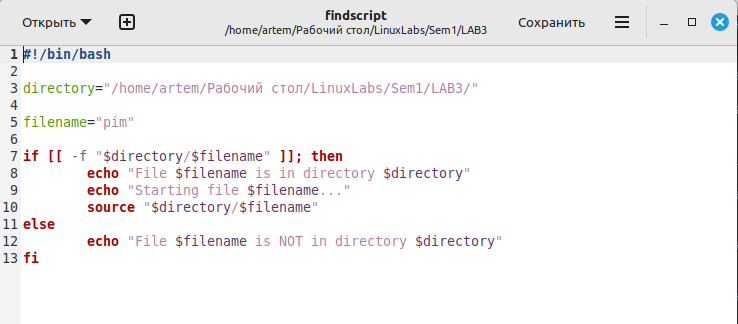


2. Проверяем результат

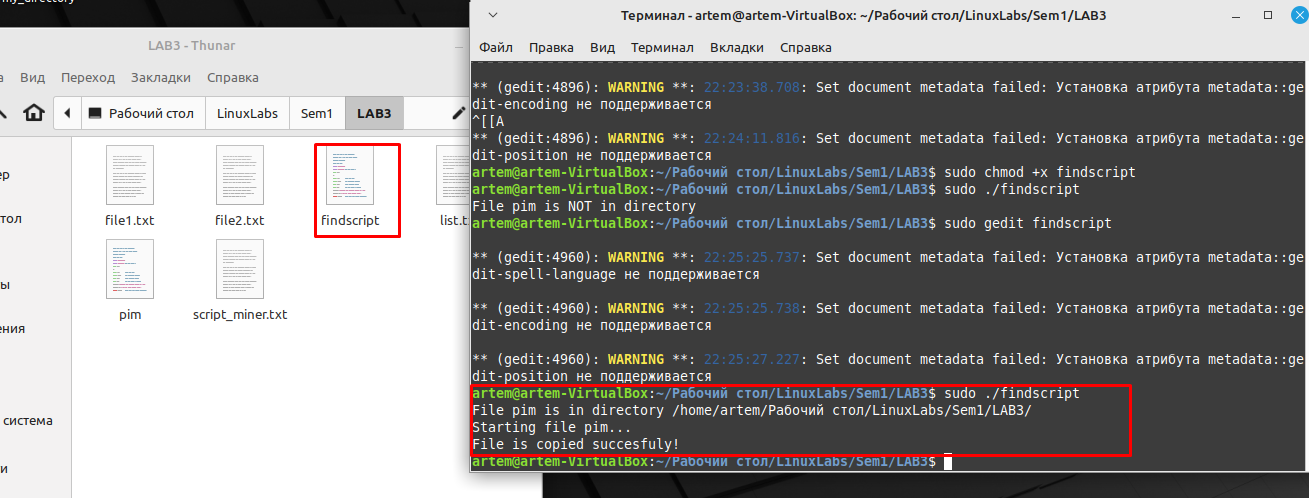


**3.** Написать скрипт, который проверяет, содержится ли скриптовый файл в директории, если да, то выполнить его.

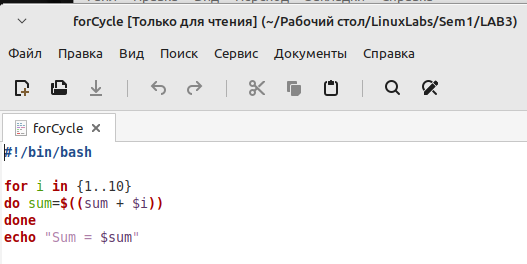
1. Пишем скрипт.



2. Запускаем и проверяем работу

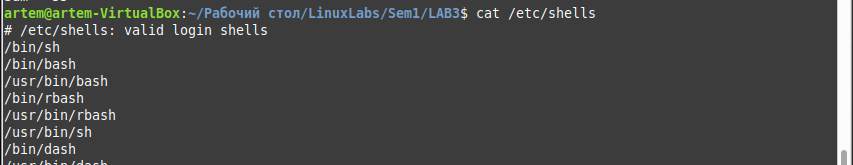


**4.** Написать скрипт, подсчитывающий сумму от 1 до 10.



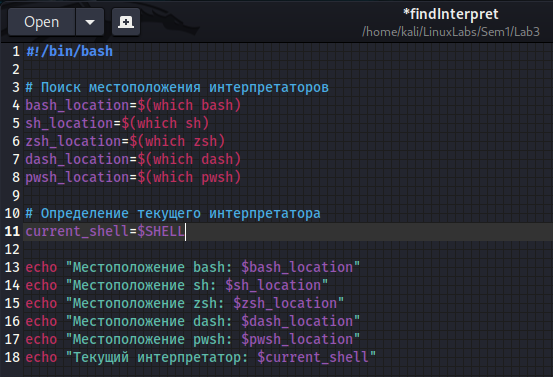


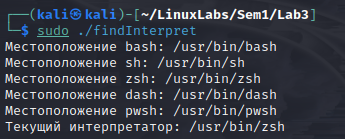
**5.** А какие командные интерпретаторы установлены в вашей системе?



sh, bash, rbach,dash

Напишите скрипт поиска местонахождения командных интерпретаторов вашей системы. Также определите какой интерпретатор используется сейчас в вашей системе?





**Задание 2.**

6. Используйте текст первого примера, заменить некоторые фамилии на фамилии членов вашей команды (например, Nutrina заменить на Stankewich).

7. Отсортируйте данные файла так, чтобы он не поменял свое собственное имя. (Решение. Нужно предварительно создать промежуточный отсортированный файл t11.txt. Затем выполнить нужные команды).

8. Подсчитайте число одинаковых слов в обоих файлах. Для подсчета числа слов в файле используйте команду: wc -w file1.

9. Написать скрипт, который создает отсортированный файл, содержащий слова из двух файлов, исключая их общую часть одинаковых слов.

10.Вывести фамилию самого молодого человека (в файле basa.txt – это второй столбец).

11.Вывести зарплату самого молодого человека (зарплата – третий столбец).

12.Вывести отсортированный список имен файлов, начальная часть имени есть script. 13.Написать скрипт, который в каждой строке файла оставляет только буквенные символы, а остальные символы выбрасывает.

Пример. Пусть содержимое файла есть:

Privet year 2022

Good bye year 2021.

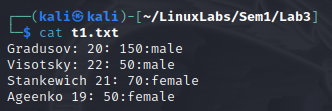
Должны получить такой файл:

Privet year

Good bye year

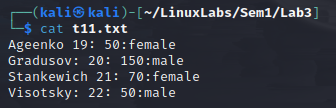
14.Вывести упорядоченный список имен файлов, в именах которых содержится символ подчеркивания, например, script\_sort.

**6.** Используйте текст первого примера, заменить некоторые фамилии на фамилии членов вашей команды (например, Nutrina заменить на Stankewich).

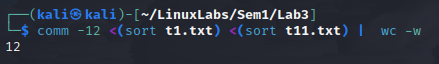


**7.** Отсортируйте данные файла так, чтобы он не поменял свое собственное имя. (Решение. Нужно предварительно создать промежуточный отсортированный файл t11.txt. Затем выполнить нужные команды).



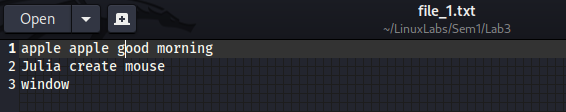


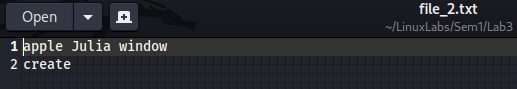
**8.** Подсчитайте число одинаковых слов в обоих файлах. Для подсчета числа слов в файле используйте команду: wc -w file1.



**9.** Написать скрипт, который создает отсортированный файл, содержащий слова из двух файлов, исключая их общую часть одинаковых слов.

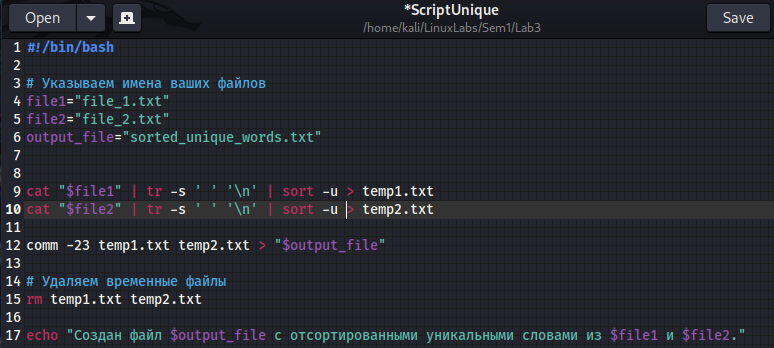
1. Создаем два файла и заполняем их.



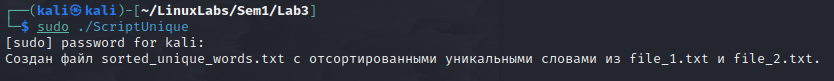


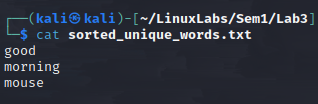
2. Пишем скрипт.

Здесь мы при помощи tr -s разделяем заменяем пробелы, но символ '\n', то есть переводим слова из одной строки на новые. Затем при помощи sort -u сортируем файл и удаляем дубликаты. И записываем результаты в два временных файла. Потом при помощи comm -23 сравниваем содержимое временных файлов и выводим только строки, которые есть только в первом файле (**-2**) и только во втором файле (**-3**).

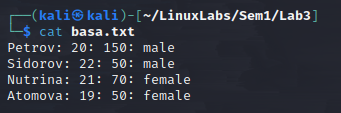


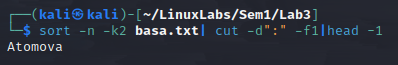
3. Проверяем результат.



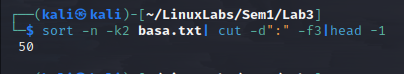


**10.** Вывести фамилию самого молодого человека (в файле basa.txt – это второй столбец).

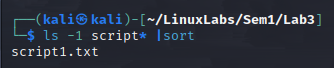




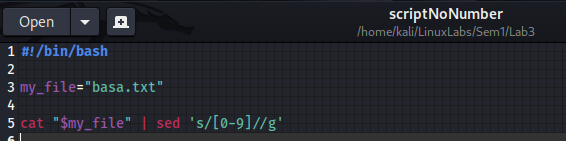
**11.** Вывести зарплату самого молодого человека (зарплата – третий столбец).

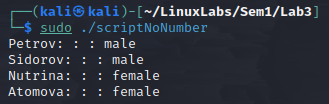


**12.** Вывести отсортированный список имен файлов, начальная часть имени есть script.

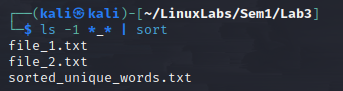


**13.** Написать скрипт, который в каждой строке файла оставляет только буквенные символы, а остальные символы выбрасывает.





**14.** Вывести упорядоченный список имен файлов, в именах которых содержится символ подчеркивания, например, script\_sort.

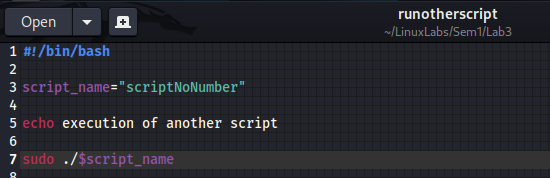


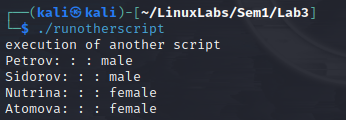
**Задание 3.**

15.Напишите скрипт запуска другого скрипта.

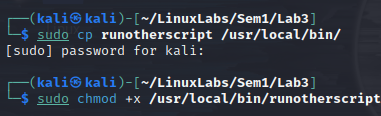
16.Расположите скрипт в одной из стантадртных директорий системы, и запустите его из домашней директории пользователя.

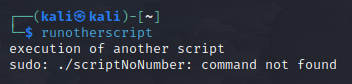
**15.** Напишите скрипт запуска другого скрипта.





**16.** Расположите скрипт в одной из стантартных директорий системы, и запустите его из домашней директории пользователя.





Контрольные вопросы:

1) Компиляция и интерпретация - это два разных способа выполнения программного кода, и они имеют существенные различия в том, как программа выполняется:

Компиляция:

1. Процесс: При компиляции исходный код программы (например, на языке C, C++, Java) преобразуется в машинный код или промежуточный байт-код. Этот процесс выполняется компилятором.

2. Выполнение: После компиляции, программа становится независимой от исходного кода и может быть выполнена многократно без повторной компиляции. Выполнение происходит непосредственно на компьютере или виртуальной машине.

3. Примеры языков: Примерами языков программирования, которые используют компиляцию, являются C, C++, Rust и др.

4. Преимущества: Компиляция может обеспечить более быстрое выполнение программы, так как код уже скомпилирован в машинный код.

Интерпретация:

1. Процесс: При интерпретации исходный код программы читается и выполняется построчно или по блокам непосредственно интерпретатором. Нет необходимости в явной компиляции.

2. Выполнение: Интерпретация происходит в реальном времени во время выполнения программы. Исходный код не преобразуется в машинный код заранее.

3. Примеры языков: Примерами языков программирования, которые используют интерпретацию, являются Python, JavaScript, Ruby и др.

4. Преимущества: Интерпретация обеспечивает большую гибкость и удобство разработки, так как изменения в коде могут быть применены мгновенно без необходимости компиляции. Она также позволяет портировать программы на разные платформы без перекомпиляции.

Пример сравнения:

Пусть у вас есть программа, написанная на языке Python. Если вы используете интерпретацию, вы можете запустить эту программу на любой платформе, где установлен интерпретатор Python, без необходимости перекомпиляции. С другой стороны, если бы программа была написана на языке C++, вам бы потребовалось компилировать ее для каждой платформы, на которой хотите ее запустить.

Каждый из этих подходов имеет свои сильные и слабые стороны, и выбор между компиляцией и интерпретацией зависит от требований к проекту и языку программирования.

2) Что рефакторинг кода?

Рефакторинг кода - это процесс изменения внутренней структуры программного кода, без изменения его внешнего поведения, с целью улучшения его читаемости, поддерживаемости и эффективности. Основная идея рефакторинга заключается в том, чтобы улучшить качество кода и сделать его более понятным и легким для работы, не изменяя при этом функциональность программы.

Рефакторинг включает в себя различные операции, такие как:

1. Переименование переменных и функций: Дать более понятные и описательные имена переменным и функциям, чтобы улучшить читаемость кода.

2. Разделение функций: Разбиение больших функций на более мелкие, что делает код более модульным и легким для понимания.

3. Удаление дублированного кода: Устранение повторяющегося кода, чтобы сократить объем кода и уменьшить вероятность ошибок.

4. Изменение структуры данных: Изменение структуры данных, если это улучшит производительность или упростит код.

5. Оптимизация алгоритмов: Улучшение алгоритмов, чтобы код работал быстрее и эффективнее.

6. Улучшение оформления кода: Приведение кода к согласованному стилю оформления, чтобы он был более читаемым и приятным для глаза.

7. Добавление комментариев: Добавление пояснительных комментариев, чтобы объяснить сложные части кода.

Рефакторинг является важной частью разработки программного обеспечения, так как помогает поддерживать и улучшать код в течение его жизненного цикла. Он уменьшает риск возникновения ошибок, облегчает совместную работу и делает код более гибким и адаптивным к изменениям в требованиях и окружении.