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* Ахраменко Н. В.

Василевский А. А.

Козлов М. А.

.

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.

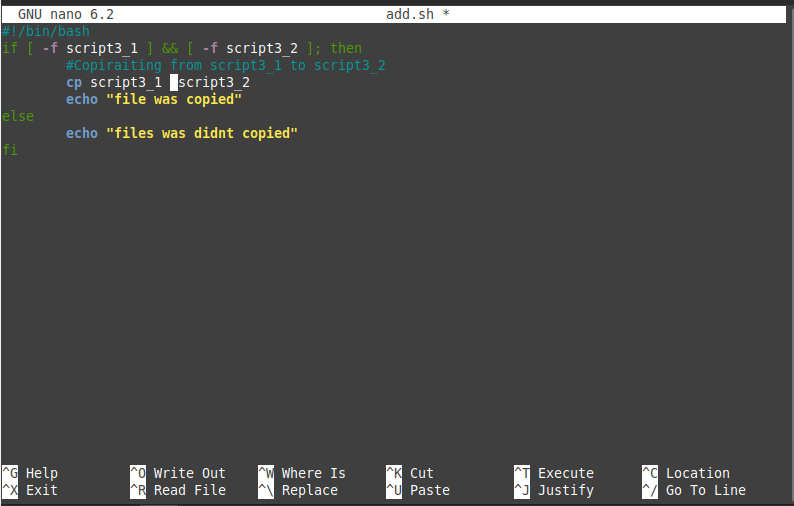
1. Создаём “sh” файл в котором будем писать скрипт.



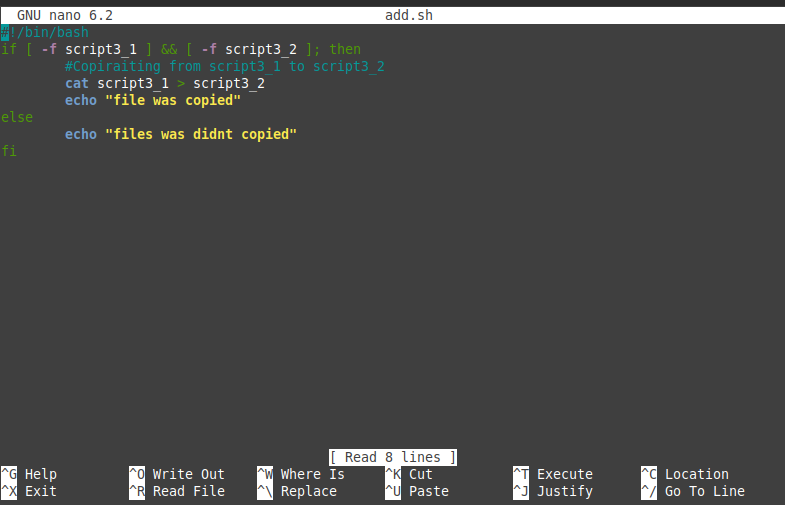
1. Заходим в файл, пишем код сценария и сохраняем файл. Для сохранения файла понадобится ввести CtrlO(сохранить файл) + Enter(Ок) + CtrlX(Выйти из файла).



1) Используя команду `cp` для копирования файла:



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



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



4. Запускаем скрипт



1. Проверяем наш второй файл:



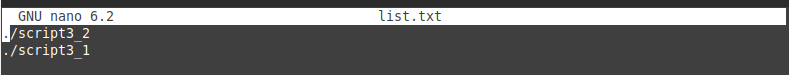


**2.** Найти все файлы, начинающиеся на слово script и записать их имена в файл list.txt.

1. Выполняем команду find

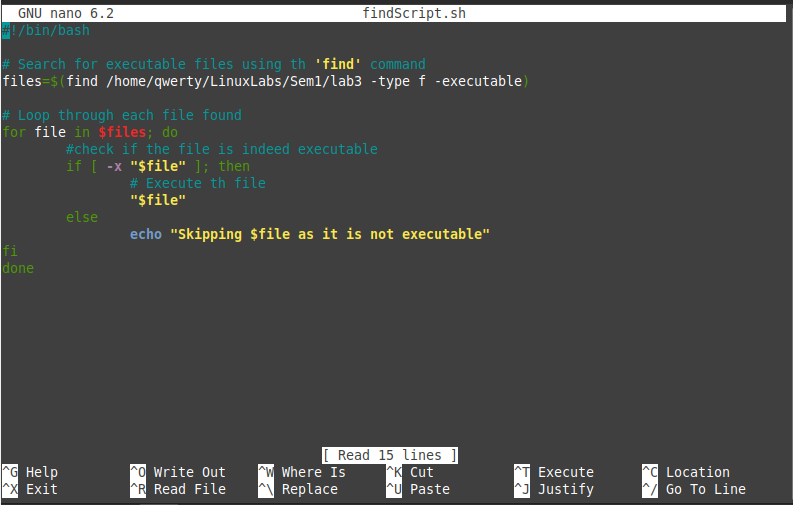


2. Проверяем результат

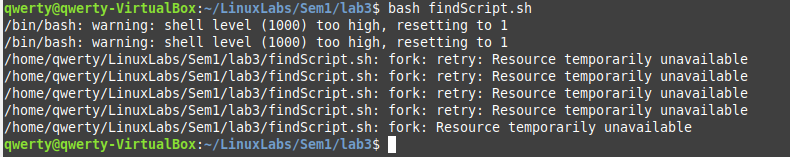


**3.** Написать скрипт, который проверяет, содержится ли скриптовый файл в директории, если да, то выполнить его.

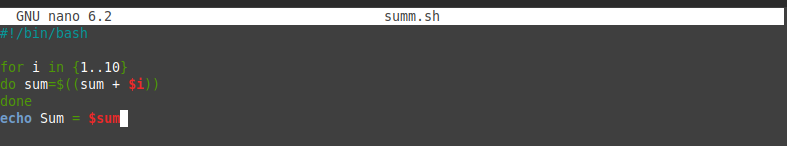
1. Пишем скрипт.



2. Запускаем и проверяем работу

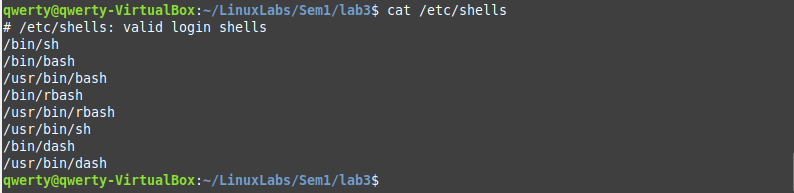


**4.** Написать скрипт, подсчитывающий сумму от 1 до 10.



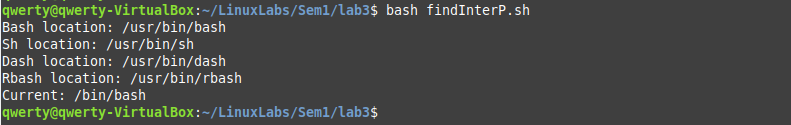


**5.** А какие командные интерпретаторы установлены в вашей системе?



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.

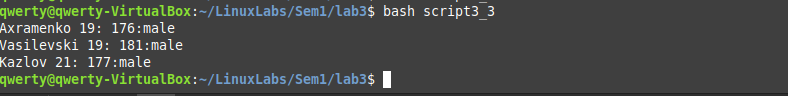
Должны получить такой файл:

Privet year

Good bye year

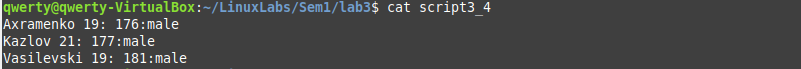
14.Вывести упорядоченный список имен файлов, в именах которых содержится символ подчеркивания, например, script\_sort.

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

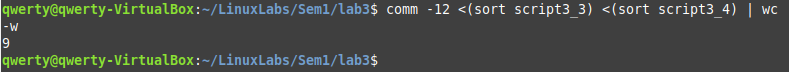


**7.** Отсортируйте данные файла так, чтобы он не поменял свое собственное имя. (Решение. Нужно предварительно создать промежуточный отсортированный файл t11.txt. Затем выполнить нужные команды).



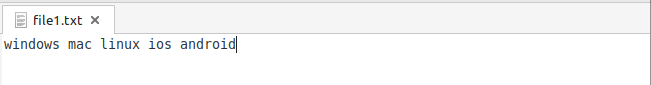


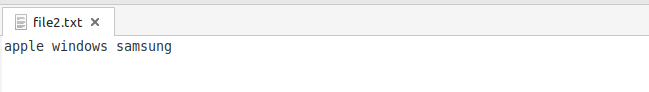
**8.** Подсчитайте число одинаковых слов в обоих файлах. Для подсчета числа слов в файле используйте команду: wc -w file1.



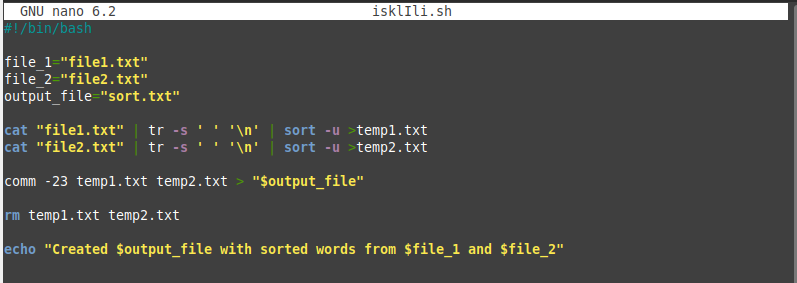
**9.** Написать скрипт, который создает отсортированный файл, содержащий слова из двух файлов, исключая их общую часть одинаковых слов.

1. Создаем два файла и заполняем их.





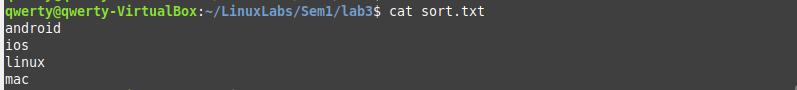
1. Пишем скрипт.



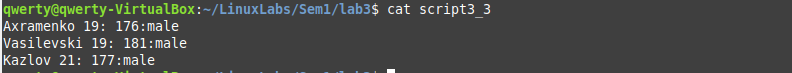
Здесь мы при помощи tr -s разделяем заменяем пробелы, на символ '\n', то есть переводим слова из одной строки на новые. Затем при помощи sort -u сортируем файл и удаляем дубликаты. И записываем результаты в два временных файла. Потом при помощи comm -23 сравниваем содержимое временных файлов и выводим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

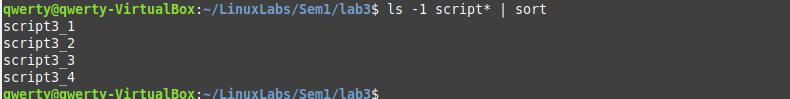




**11.** Вывести зарплату самого молодого человека (зарплата – третий столбец).

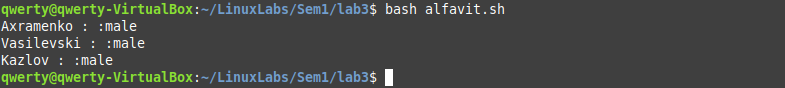


**12.** Вывести отсортированный список имен файлов, начальная часть имени есть script.

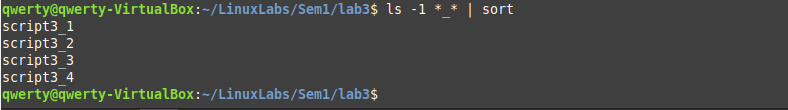


**13.** Написать скрипт, который в каждой строке файла оставляет только буквенные символы, а остальные символы выбрасывает.





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

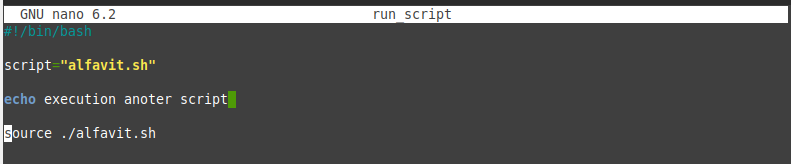


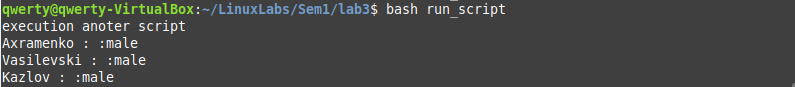
**Задание 3.**

15.Напишите скрипт запуска другого скрипта.

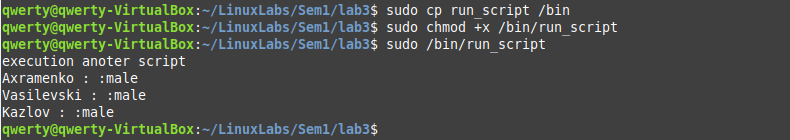
16.Расположите скрипт в одной из стандартных директорий системы, и запустите его из домашней директории пользователя.

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

****



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



**Контрольные вопросы:**

**Скрипты или сценарии командной оболочки**

**1) Что такое *область видимости* переменных? Какие типы области видимости переменных существуют?**

Область видимости переменных - это часть программы, в которой переменная может быть доступна и использована. Она определяет место, где переменная определена и может быть обращена к ней.

Существует несколько типов областей видимости переменных:

1. Глобальная область видимости: переменная объявлена вне всех функций и классов и может быть использована по всему коду программы.

2. Локальная область видимости: переменная объявлена внутри функции или блока кода и доступна только в пределах этой функции или блока. Например, если переменная объявлена внутри цикла или условия, она видна и доступна только внутри этого цикла или условия.

3. Параметрическая область видимости: переменные, переданные в функцию в качестве параметров, имеют область видимости, ограниченную этой функцией. Они видны только внутри функции и недоступны вне её.

4. Область видимости класса: переменные, объявленные внутри класса, могут быть доступны только для объектов этого класса. Они обычно называются полями или свойствами класса.

Каждый тип области видимости имеет свои правила доступа и использования переменных, что помогает управлять видимостью и избегать конфликтов и неоднозначности при работе с переменными.

**Какие существуют типы окон примитивного графического интерфейса Linux-скриптов?**

В Linux существует несколько примитивных графических интерфейсов и оконных систем, которые могут использоваться в скриптах. Некоторые из них включают:

1. X Window System (X11): Он является стандартной оконной системой в большинстве дистрибутивов Linux. Он предоставляет базовые функции для создания и управления окнами, а также обеспечивает вывод графики.

2. Xfce: Это легковесная оконная система, которая предлагает минималистичный интерфейс с низким потреблением ресурсов. Она широко используется в легких дистрибутивах Linux.

3. LXDE: Он также является легковесной оконной системой, предназначенной для использования на маломощных системах или виртуальных машинах. Он предлагает простой и интуитивно понятный интерфейс.

4. Openbox: Это менеджер окон с открытым исходным кодом, предоставляющий минимальный функционал с возможностью настройки. Он позволяет пользователям создавать собственные оконные раскладки и горячие клавиши.

5. Fluxbox: Он также является легковесной и быстрой оконной системой, основанной на Openbox. Он предлагает высокую настраиваемость и расширяемость.

Это лишь некоторые из примеров оконных систем и менеджеров окон, доступных в Linux для использования в скриптах. Выбор конкретной оконной системы в значительной степени зависит от ваших предпочтений, требований и особенностей вашего проекта.