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 10701321

Черепковский М.В.

Принял**:** Давыденко Н.В.

Минск 2023

# Лабораторная работа № 3. Командный язык и скрипты Shell

# 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 интерпретато-

ров вашей системы. Также определите какой интерпретатор использу-

ется сейчас в вашей системе?

**Решение**

1. A) Для копирования можно использовать команду cp file1 file2, которая перенесёт текст из file1 в file2.

Б) Для копирования можно использовать команду cat file1 > file2, которая перенесёт текст из file1 в file2.

1. Для выполнения можно использовать команду find / -type f -name "script\*" > list

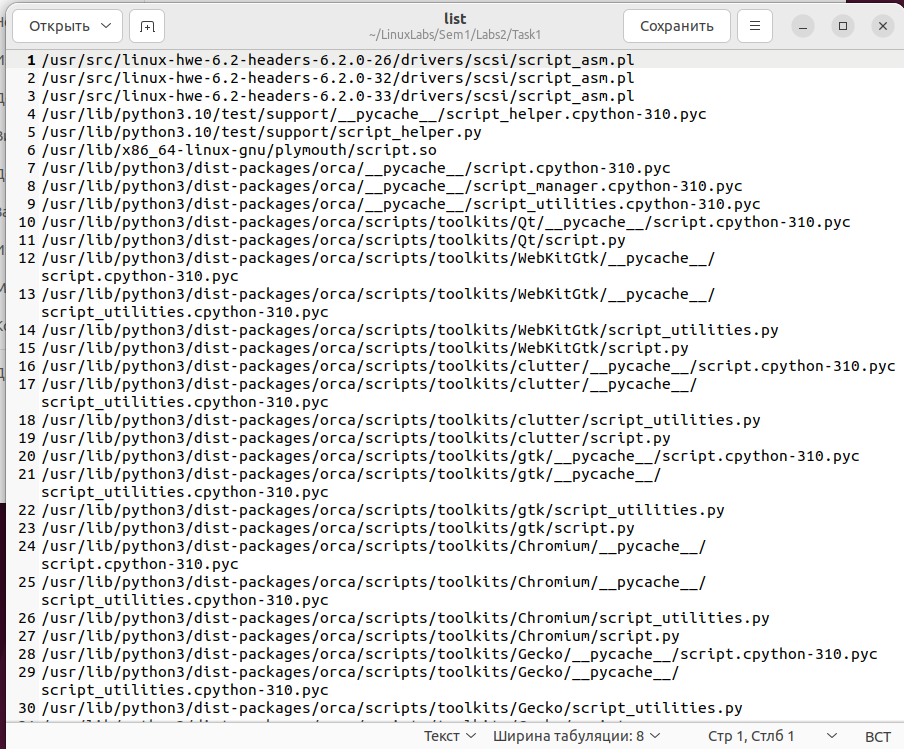
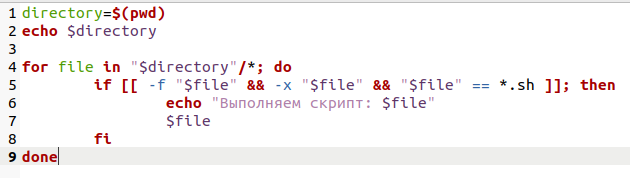


Рис. 1. Результат поис№ка файлов, начинающихся на script

1. Для выполнения задания, напишем скрипт, который проходится по всем файлам в папке, проверяет, являются ли они файлами, являются ли они исполняемыми и имеют расширение .sh, ексли все 3 условия верные, то выполнятеся скрипт



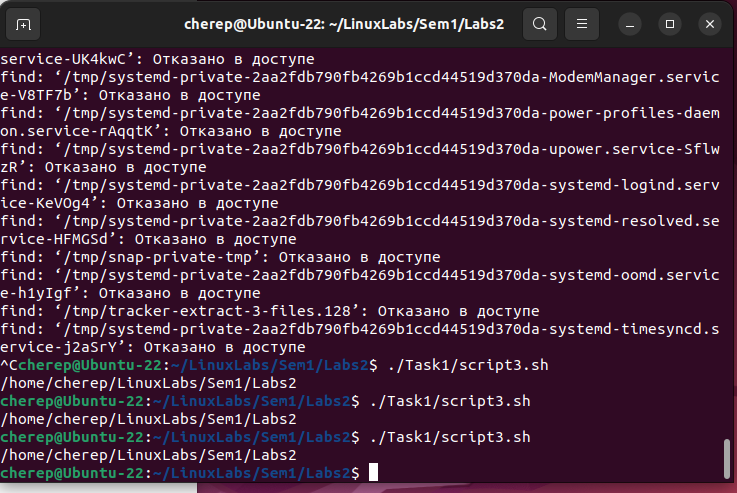


Рис. 2. Результат выполнения задания 3

1. Для выполнения задания используем цикл for

sum=0

for ((i=1; i<=10; i++)); do

sum=$((sum + i))

done

echo "Сумма чисел: $sum"



Рис.3. Сумма чисел от 1 до 10

1. Для выполнения задания напишем скрипт

echo "Путь к bash: $(which bash)"

echo "Путь к sh: $(which sh)"

echo "Путь к dash: $(which dash)"

echo "Текущий интерпретатор: $(ps -p $$ -o comm=)"

Команда which выводит путь к интерпритатору. $(ps -p $$ -o comm=) берёт текущий процесс скрипта и получает информацию о том, с помощью какого интепретатора был выполнен данный скрипт.

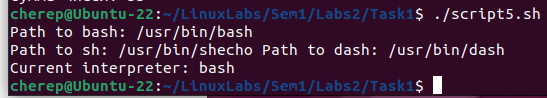


Рис.4. Пути к интепретаторам

**Задание 2**

1. Используйте текст первого примера, заменить некоторые фамилии на

фамилии членов вашей команды (например, Nutrina заменить на

Stankewich).

2. Отсортируйте данные файла так, чтобы он не поменял свое собствен-

ное имя. (Решение. Нужно предварительно создать промежуточный от-

сортированный файл t11.txt. Затем выполнить нужные команды).

3. Подсчитайте число одинаковых слов в обоих файлах. Для подсчета

числа слов в файле используйте команду: wc -w file1.

4. Написать скрипт, который создает отсортированный файл, содержащий

слова из двух файлов, исключая их общую часть одинаковых слов.

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

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

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

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

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

**Решение**

1. Создадим файл peoples

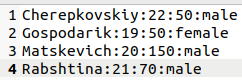


Рис.5. Файл peoples

1. Для выполнения задания напишем скрипт, который сортирует данные и копирует их в файл peoples1, а затем копирует содержание файла peoples1 в файл peoples

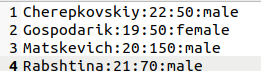


Рис.6. Результат выполнения скрипта

1. Для выполнения задания напишем скрипт, который объединяет тексты двух файлов, объединяет и сортирует их, затем выбирает только дубликаты и считает их количество

sort peoples peoples1 | uniq -d > common

total\_words=$(wc -w < common)

echo "Общее количество одинаковых слов: $total\_words"

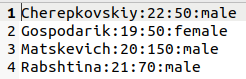


Рис.7. Общие слова



Рис.8. Результат выполнения

1. Для выполнения задания напишем скрипт, который сортирует 2 файла, затем находит разницу между ними и вставляет в новый файл.

sort peoples > sorted\_peoples

sort peoples1 > sorted\_peoples1

comm -3 sorted\_peoples sorted\_peoples1 > uncommon

echo "Файл с уникальными словами создан"



Рис.9. Несовпадающие слова

Команда comm используется для сравнения двух отсортированных файлов построчно и вывода результатов сравнения. В данном случае команда comm используется с опцией -3, что означает "подавить вывод столбцов, которые присутствуют в обоих файлах".

1. Для выполнения задания напишем скрипт, который сортирует данные по второму столбцу в порядке возрастания, возьмёт первую строку.

sort -t: -k2 -n peoples | head –n1

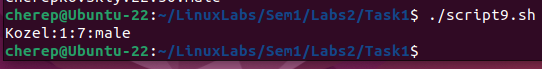


Рис.10. Фамилия самого молодого человека

sort -t: -k2 -n peoples: Сначала она сортирует содержимое файла peoples с использованием двоеточия (:) в качестве разделителя полей (-t:). Опция -k2 указывает сортировать по второму полю (в данном случае, числовому), и опция -n указывает сортировать числа как числа, а не как строки.

head -n1: Затем команда head -n1 используется для выбора первой строки (наименьшего числа) из отсортированного списка.

1. Для выполнения задания напишем скрипт, который сортирует данные по второму столбцу в порядке возрастания, возьмёт первую строку, поделит на ячейки и возьмёт 3



Рис.11. Заработок самого молодого человека

awk -F: '{print $3}': Наконец, команда awk используется для разбора строки с использованием двоеточия (:) как разделителя полей (опция -F:) и выводит первое поле (в данном случае, имя) этой строки.

1. Получим все файлы в папке, выбирем те, которые начинаются на “script”, и сортируем их по названию.

ls | grep '^script' | sort –t’t’ –k2n

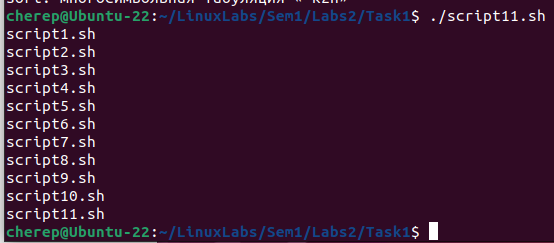


Рис.12. Результат выполнения

ls используется для вывода списка файлов и директорий в текущей директории.

grep '^script' фильтрует только те строки, которые начинаются с "script".

sort -t' ' -k2n выполняет сортировку строк, разделенных пробелами (по умолчанию), по второму полю (-k2) как числовому значению (-n). Это обеспечит правильную сортировку чисел.

1. Для выполнения задания напишем скрипт, который с помощью sed и регулярного выражения находит все небуквенные символы и заменяет их на пустоту и результат записывает в файл new\_text  
   sed 's/[^[:alpha:]]//g' peoples.txt > new\_text

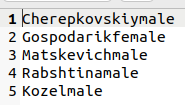


Рис.13. Текст с отсутствием небуквенных символов

sed может найти текстовые строки, соответствующие определенному шаблону, и заменить их другим текстом.

1. Для выполнения задания напишем скрипт, который получает все файлы из директории, выбирает из них файлы, имеющие “\_” и сортирует их

ls | grep '\_' | sort

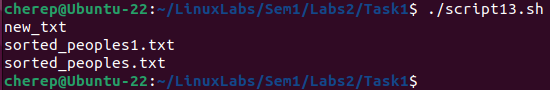


Рис.14. Результат выполнения

**Задание 3**

1. Напишите скрипт запуска другого скрипта.

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

**Решение**

1. Для запуска одного скрипта из другого в командной оболочке (например, в Bash), вы можете использовать команду bash или ./, в зависимости от того, какой способ вам удобнее.

bash script11.sh

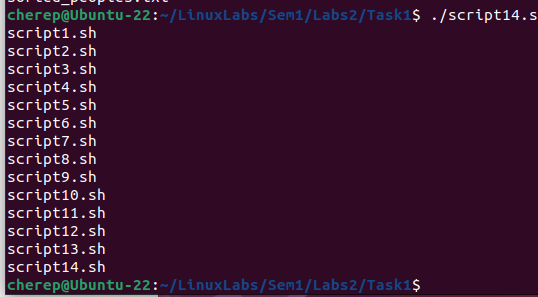


Рис.15. Результат выполнения



Рис.16. Результат выполнения

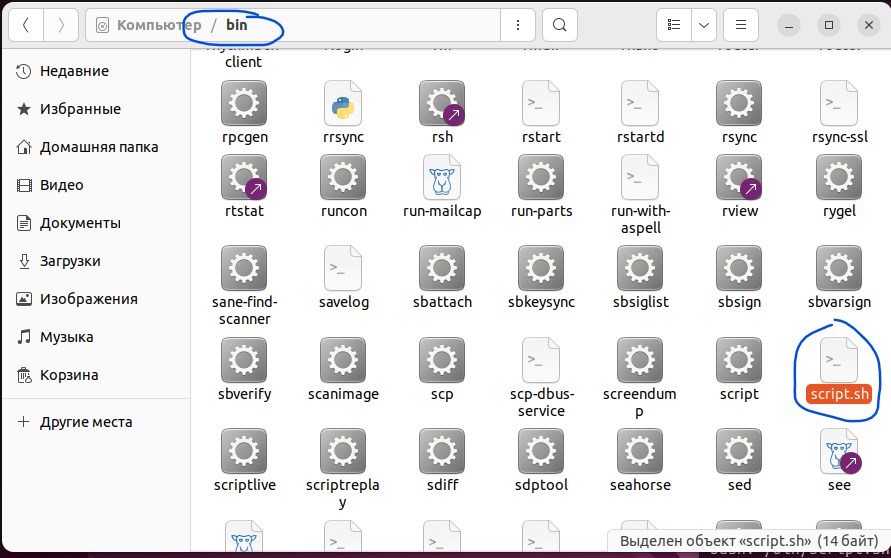


Рис.17. Результат выполнения

Вывод*:* В ходе выполнения лабораторной работы изучили основные команды, для написания скриптов.

**Контрольные вопросы:**

1. Что такое скрипт или сценарий командной оболочки, для чего они используются?

Скрипт или сценарий командной оболочки (иногда также называемый шелл-скриптом) - это текстовый файл, который содержит последовательность команд для выполнения в командной строке или интерпретаторе командной оболочки. Эти команды обычно записываются на языке командной оболочки, таком как Bash, sh, zsh, PowerShell и других, и могут выполнять различные задачи и автоматизировать операции.

Основные цели использования скриптов командной оболочки:

1. \*\*Автоматизация задач\*\*: С помощью скриптов можно автоматизировать выполнение серии команд или задач, что позволяет значительно упростить рутинные операции. Например, вы можете создать скрипт для резервного копирования файлов, автоматической обработки данных или выполнения регулярных системных задач.

2. \*\*Пакетная обработка данных\*\*: Скрипты могут использоваться для обработки больших объемов данных или файлов, применяя к ним определенные операции или фильтры.

3. \*\*Настройка и конфигурирование\*\*: Скрипты позволяют изменять настройки и конфигурации системы или программ, что упрощает задачи администрирования и управления.

4. \*\*Создание пользовательских приложений\*\*: С помощью скриптов можно создавать простые командной строки приложения или утилиты.

5. \*\*Отладка и анализ\*\*: Скрипты часто используются для отладки, анализа и регистрации информации о системе и приложениях.

6. \*\*Управление файловой системой\*\*: Скрипты позволяют создавать, копировать, перемещать, изменять права доступа и удалять файлы и директории.

Различные командные оболочки имеют свои собственные синтаксисы и возможности для написания скриптов, но основная идея остается общей - использование текстовых файлов для автоматизации и управления операциями в системе.

1. Назовите три основновополагающие концепции операционных систем и системного программирования?

Три основополагающие концепции операционных систем и системного программирования включают в себя:

1. \*\*Процессы\*\*: Процесс является основной единицей выполнения в операционной системе. Он представляет собой программу во время выполнения и включает в себя код, данные и ресурсы, необходимые для выполнения программы. Операционная система управляет созданием, управлением и завершением процессов, а также распределением ресурсов между ними.

2. \*\*Память и управление ресурсами\*\*: Операционная система управляет доступом к памяти и другим ресурсам компьютера. Она отвечает за управление выделением и освобождением памяти, управление процессами, управление файловой системой и другими ресурсами, чтобы обеспечить эффективное и безопасное выполнение задач.

3. \*\*Многозадачность и многопользовательский доступ\*\*: Операционная система поддерживает многозадачность, что означает способность выполнять одновременно несколько процессов. Она также обеспечивает многопользовательский доступ, позволяя нескольким пользователям одновременно использовать компьютер и ресурсы. Это особенно важно для современных многопользовательских систем, таких как серверы и многозадачные рабочие станции.

Эти концепции являются фундаментальными для операционных систем и системного программирования и обеспечивают эффективное и надежное функционирование компьютерных систем.