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

Господарик П. 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 интерпретаторов вашей системы. Также определите какой интерпретатор используется сейчас в вашей системе?

**Решение**

1. V1) Для копирования можно использовать команду cat file1 > file2, которая перенесёт текст из file1 в file2.

V2) Для копирования можно использовать команду cp file1 file2, которая перенесёт текст из file1 в file2.

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

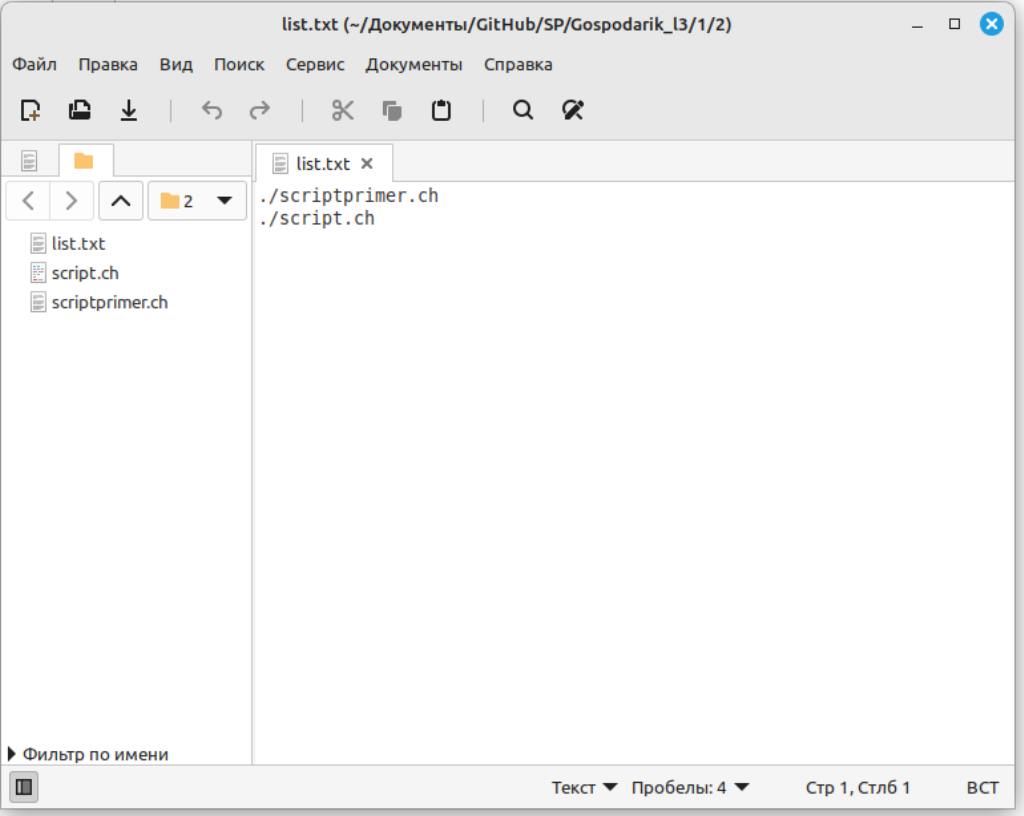


Рис. 1. Результат поиска файлов, начинающихся на script

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

directory="$PWD/dir"

files\_list=$(find "$directory" -type f)

for file in "$directory"/\*; do

if [[ -x "$file" && "$file" == \*.sh ]]; then

echo "Cкрипт: $file"

$file

fi

done

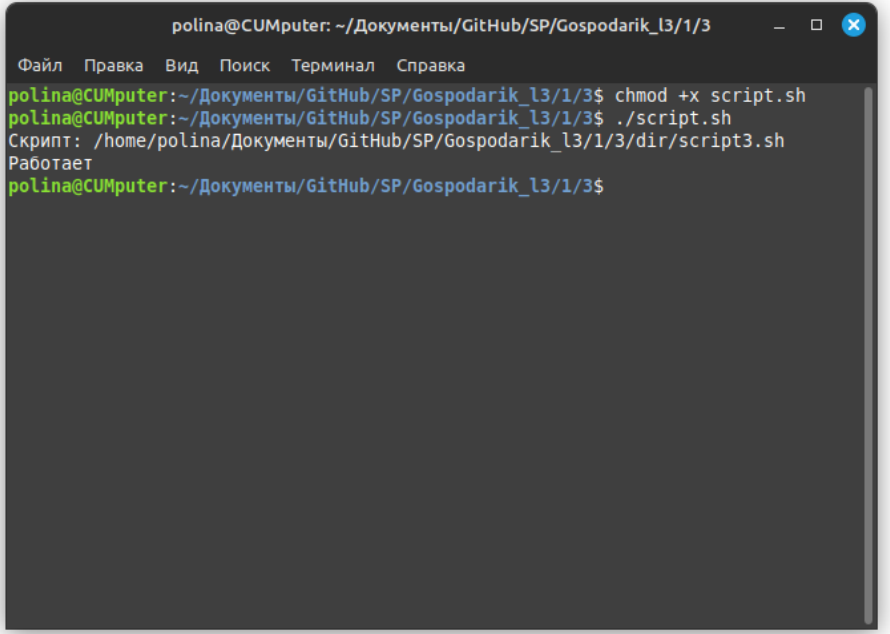


Рис. 2. Результат выполнения задания 3

1. Для выполнения задания используем цикл for

#!/bin/bash

sum=0

for i in {1..10}; do

sum=$((sum + i))

done

echo "Сумма чисел от 1 до 10: $sum"

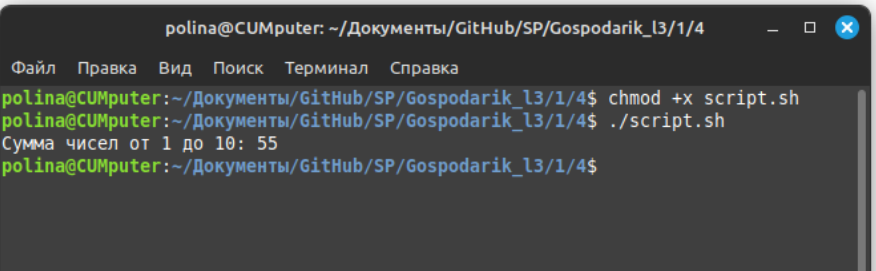


Рис.3. Сумма чисел от 1 до 10

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.

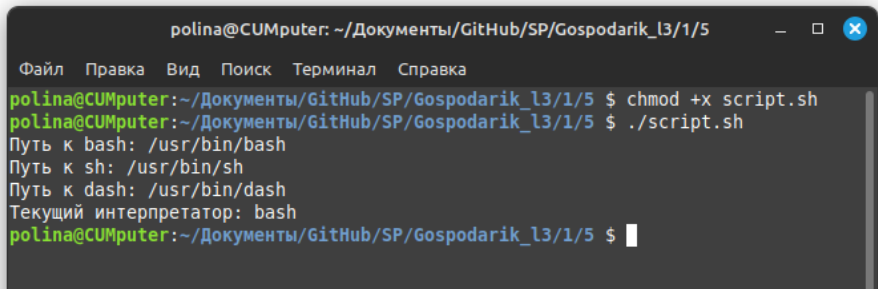


Рис.4. Пути к интепретаторам

**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

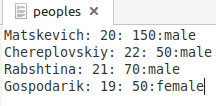


Рис.5. Файл peoples

1. Для выполнения задания напишем скрипт, который сортирует данные и копирует их в файл people1, а затем копирует содержание файла people1 в файл people

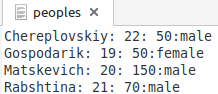


Рис.6. Результат выполнения скрипта

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"

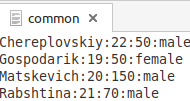


Рис.7. Общие слова

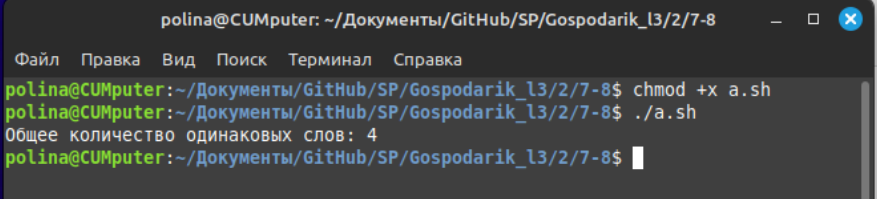


Рис.8. Результат выполнения

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"

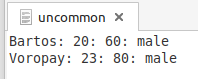


Рис.9. Несовпадающие слова

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

sort -t: -k2 -n people | head -n1 | awk -F: '{print $1}'

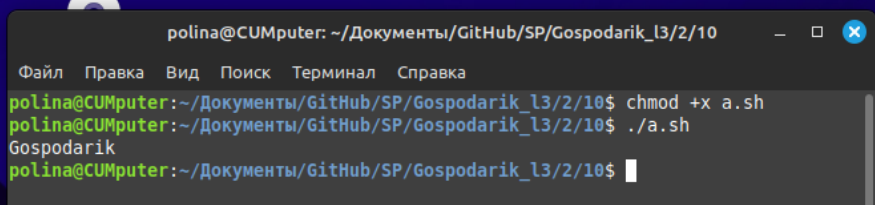


Рис.10. Фамилия самого молодого человека

1. Для выполнения задания изменим предыдущий скрипт, изменив {print $1}, на {print $3}, который возьмёт 3 ячейку

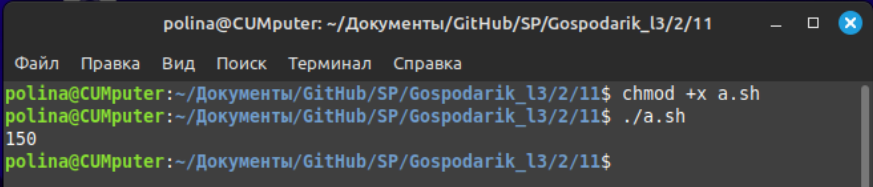


Рис.11. Заработок самого молодого человека

1. который получает все файлы в папке, выбирает те, который начинаются на “script”, и сортирует их по названию

ls | grep '^script' | sort

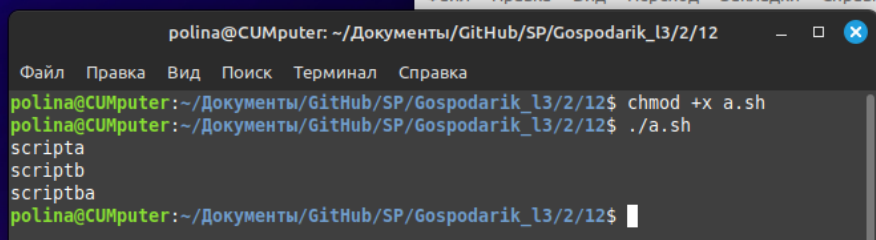


Рис.12. Результат выполнения

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' text > new\_text

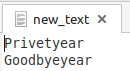


Рис.13. Текст с отсутствием небуквенных символов

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

ls | grep '\_' | sort

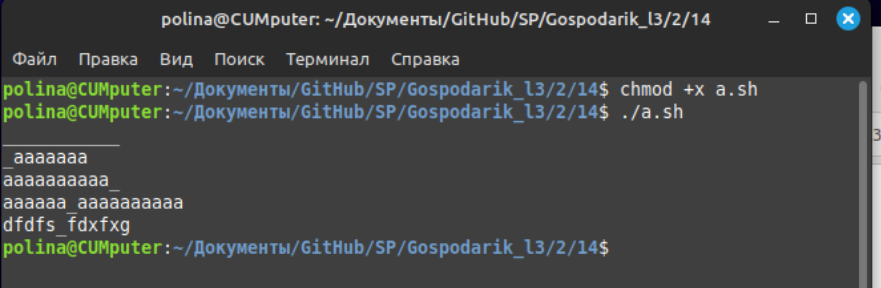


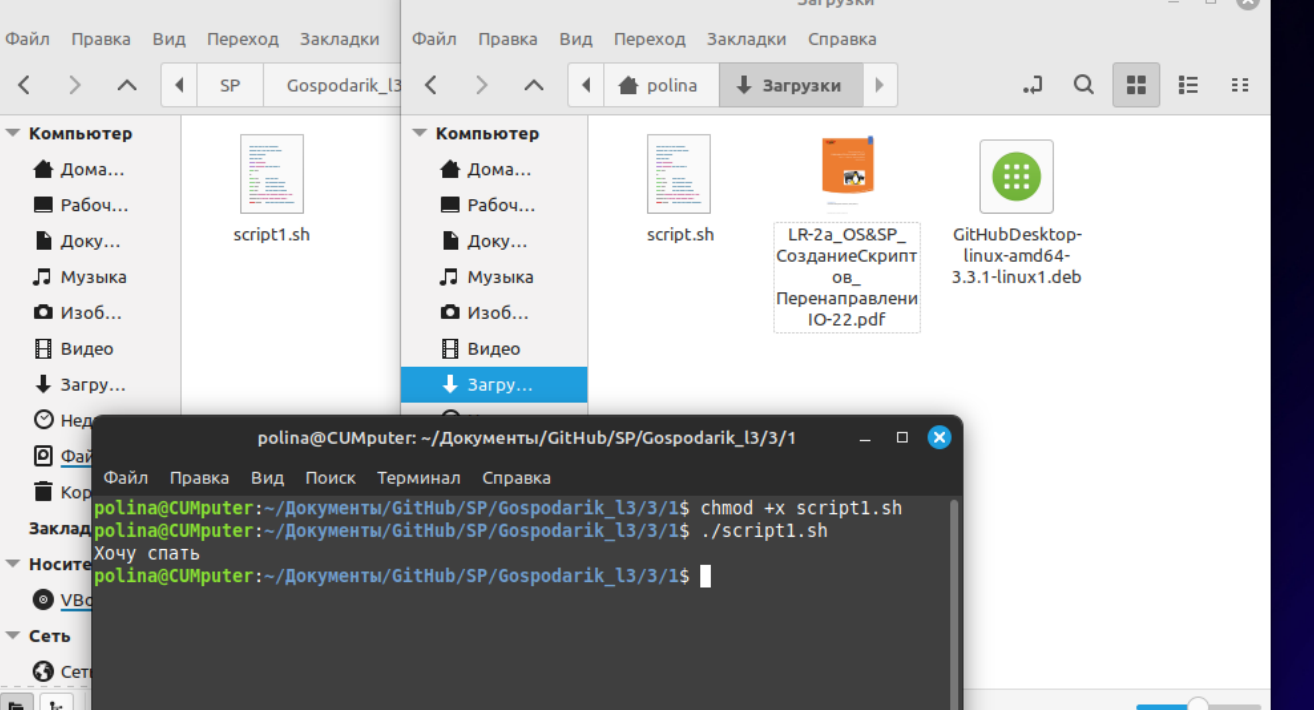
Рис.14. Результат выполнения

**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**



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

**Контрольные вопросы:**

**Какими провами доступа должны бить наделены файлы скриптов?**

Файлы скриптов, чтобы быть исполняемыми в Unix-подобных системах (таких как Linux), должны иметь соответствующие права доступа для выполнения. Обычно используется комбинация разрешений rwx (чтение, запись и выполнение). Вот что они означают

**r (read):** Позволяет читать (просматривать) содержимое файла.

**w (write):** Позволяет записывать (редактировать или изменять) содержимое файла.

**x (execute):** Позволяет выполнять файл в качестве программы или скрипта.

Чтобы установить соответствующие права на выполнение для файлов скриптов, вы можете использовать команду chmod. Например, чтобы разрешить чтение, запись и выполнение для владельца файла и разрешить чтение и выполнение для других пользователей, используйте:

bash

Copy code

chmod u+rwx script.sh

chmod go+rx script.sh

Эти команды устанавливают следующие права доступа:

u+rwx: Разрешает чтение, запись и выполнение для владельца файла (user).

go+rx: Разрешает чтение и выполнение для группы (group) и других пользователей (others).

После применения этих прав, скрипт можно будет запустить, используя команду ./script.sh из той директории, в которой он находится, или через полный путь к файлу.

Убедитесь, что вы предоставляете минимально необходимые права доступа, чтобы обеспечить безопасность вашей системы.

**Что такое потоки данных, и какие стандартные потоки данных существуют в системе? В какой системной директории располагаются их файлы?**

Потоки данных (или стандартные потоки) - это механизм передачи данных между программами или между программой и системой операций в Unix-подобных системах. Существуют три стандартных потока данных:

Стандартный ввод (stdin): Это поток, через который программа получает вводные данные. По умолчанию стандартный ввод связан с клавиатурой, но его можно перенаправить на чтение данных из файла или другого источника.

Стандартный вывод (stdout): Это поток, через который программа выводит данные. По умолчанию стандартный вывод связан с экраном (консолью), но его также можно перенаправить для записи данных в файл или другой приемник.

Стандартный вывод ошибок (stderr): Это поток, используемый для вывода сообщений об ошибках и диагностических сообщений. По умолчанию он также связан с экраном, но его можно перенаправить, чтобы сохранить сообщения об ошибках в отдельный файл.

В Unix-подобных системах файлы, связанные со стандартными потоками, находятся в специальных системных директориях:

Стандартный ввод (stdin): Связан с файлом /dev/stdin.

Стандартный вывод (stdout): Связан с файлом /dev/stdout.

Стандартный вывод ошибок (stderr): Связан с файлом /dev/stderr.

Когда вы используете команду в командной строке или запускаете программу, эти потоки данных обычно связаны с соответствующими файлами в /dev. Например, если вы выполняете команду ls > output.txt, стандартный вывод команды ls будет перенаправлен в файл output.txt. Здесь > используется для перенаправления стандартного вывода в файл.

Таким образом, стандартные потоки данных являются важной частью взаимодействия программ в Unix-подобных системах, и их перенаправление позволяет более гибко управлять вводом, выводом и сообщениями об ошибках в системе.