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 № 2а**

по дисциплине: «Системное программирование»

на тему: «Командный язык и скрипты SHELL.»

Выполнила: ст. гр. 10702221 Баяманова А.

Проверил: Давыденко Н.В.

Минск 2023

**ЛАБОРАТОРНАЯ РАБОТА #2.1**

**Командный язык и скрипты SHELL: Часть 1 – Скрипты и регулярные выражения**

**Цель работы:**

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

**Задание 1**

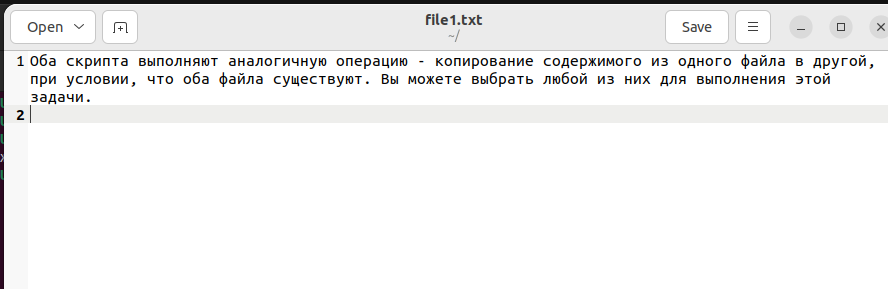
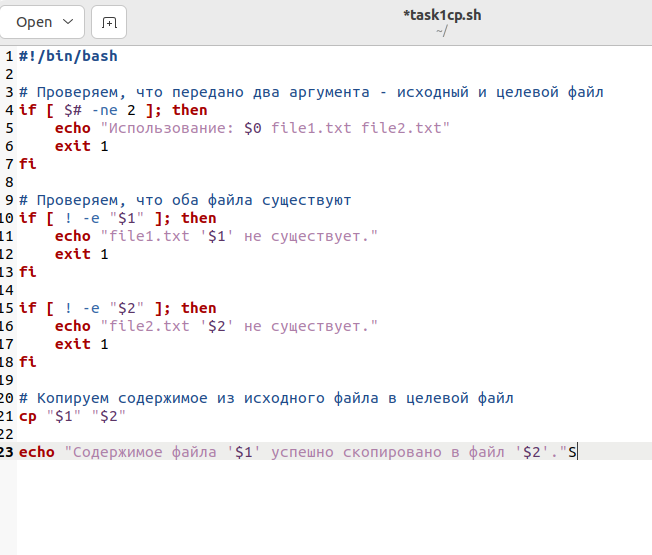
**1. Написать скрипт, который записывает содержимое одного файла в**

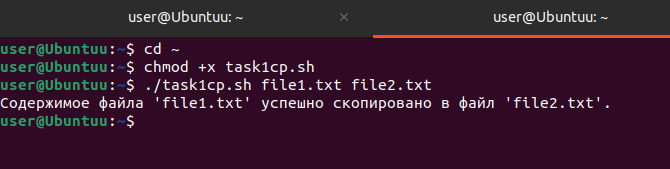
**другой и при этом оба файла существуют до операции. Дать два разных**

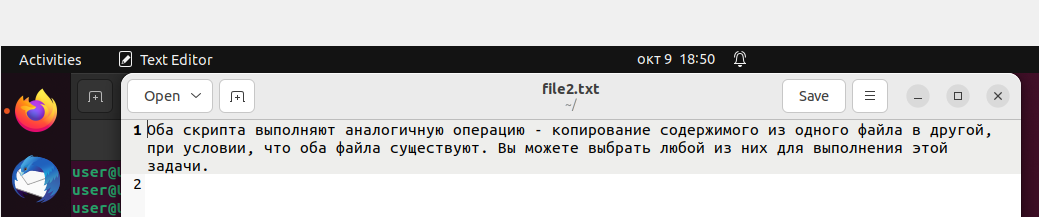
**варианта выполнения.**

Вот два разных варианта скрипта на Linux, которые копируют содержимое одного файла в другой, при условии, что оба файла существуют до операции:

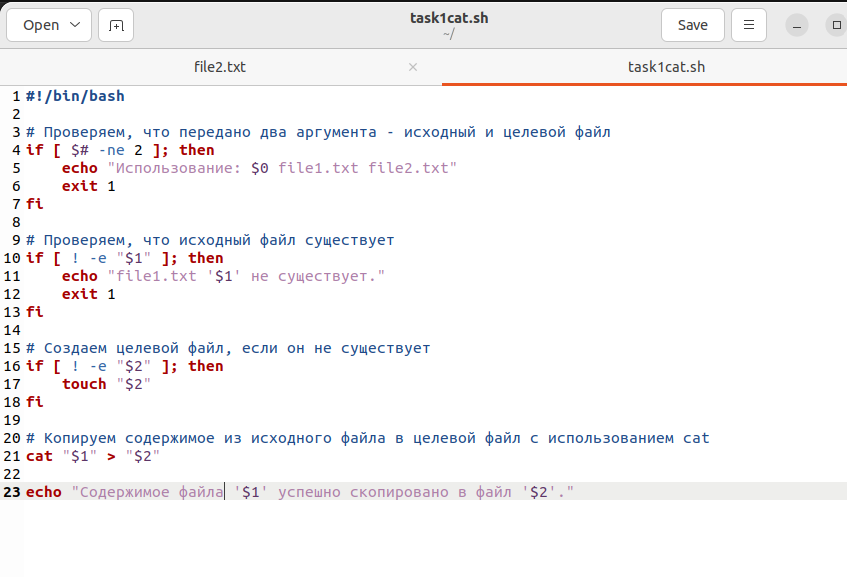
Вариант 1: Использование команды `cp`:

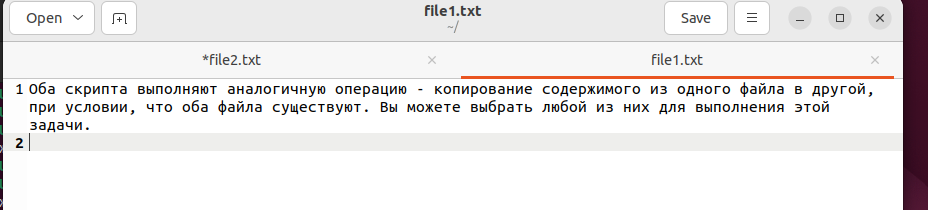


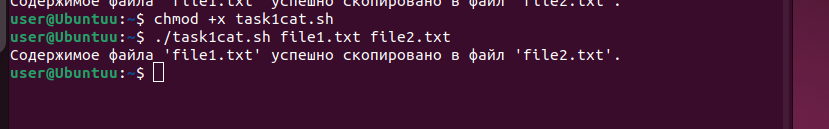


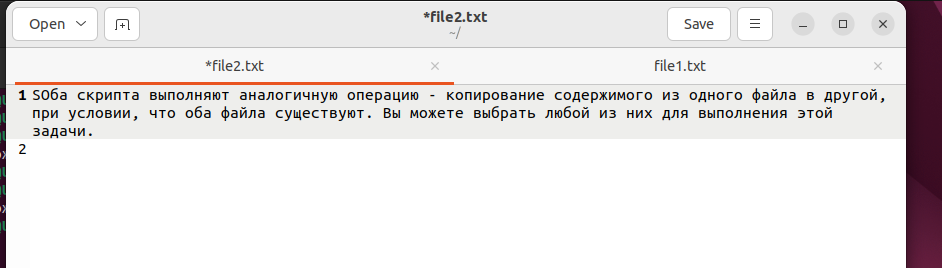


Вариант 2: Использование оператора `cat`:







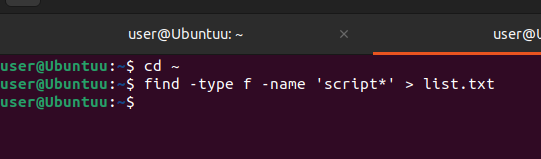


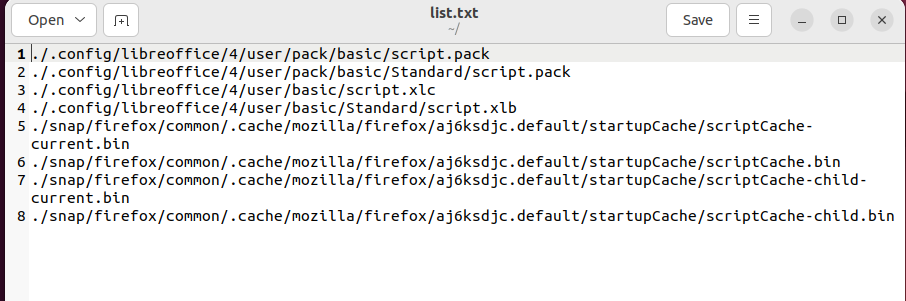
Оба скрипта выполняют аналогичную операцию - копирование содержимого из одного файла в другой, при условии, что оба файла существуют.

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

**файл list.txt.**

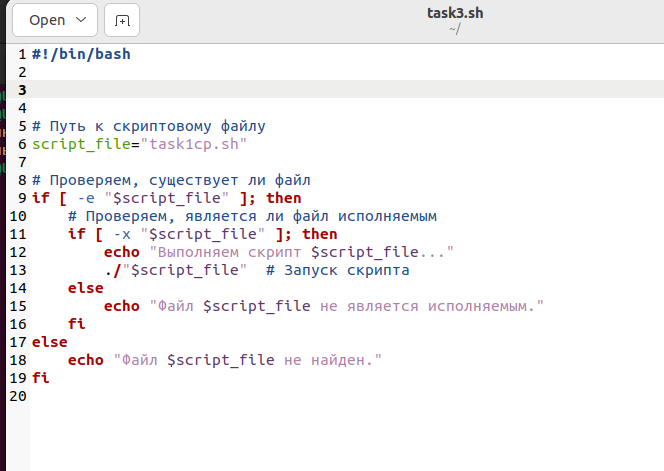
Чтобы найти все файлы, начинающиеся с слова "script" и записать их имена в файл list.txt, можно использовать команду find в сочетании с grep для фильтрации имен файлов.

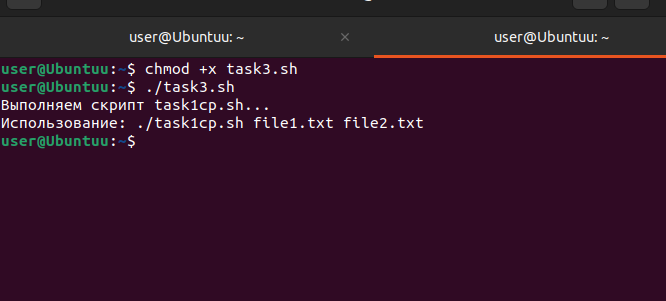


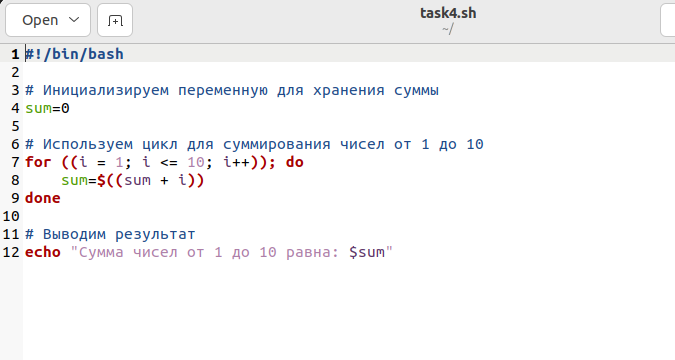
ы

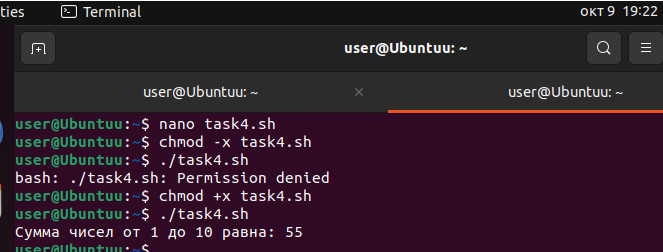
**3. Написать скрипт, который проверяет, содержится ли скриптовый файл в**

**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?**

в большинстве современных систем Linux и Unix, а также macOS, обычно установлены несколько стандартных командных интерпретаторов, такие как:

1. \*\*Bash (Bourne-Again SHell)\*\* - один из наиболее распространенных и используемых командных интерпретаторов.

2. \*\*sh (Bourne Shell)\*\* - классический интерпретатор, который был предшественником Bash.

3. \*\*dash (Debian Almquist Shell)\*\* - легковесный интерпретатор, используемый по умолчанию в Debian.

4. \*\*zsh (Z Shell)\*\* - мощный и расширяемый интерпретатор с продвинутыми функциями.

5. \*\*fish (Friendly Interactive SHell)\*\* - интерпретатор с удобным синтаксисом и автодополнением.

6. \*\*csh (C Shell)\*\* - интерпретатор, созданный на основе языка программирования C.

7. \*\*tcsh (Tenex C Shell)\*\* - расширение C Shell с дополнительными функциями.

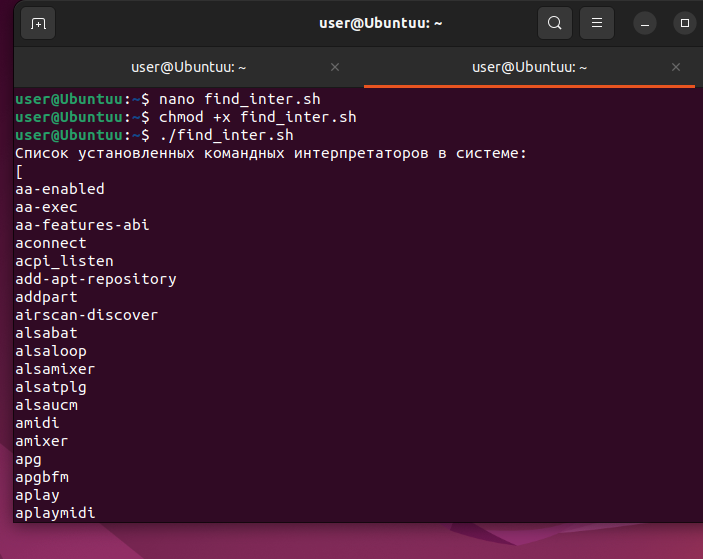
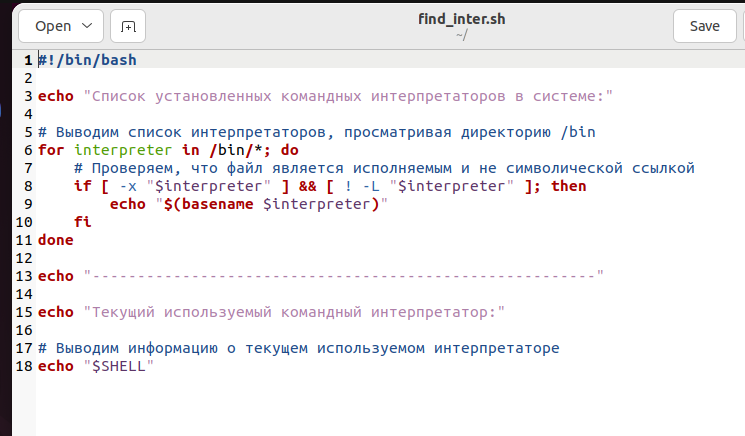
Чтобы узнать, какие именно интерпретаторы установлены на вашей системе, можно выполнить команду `ls` в каталоге `/bin`:

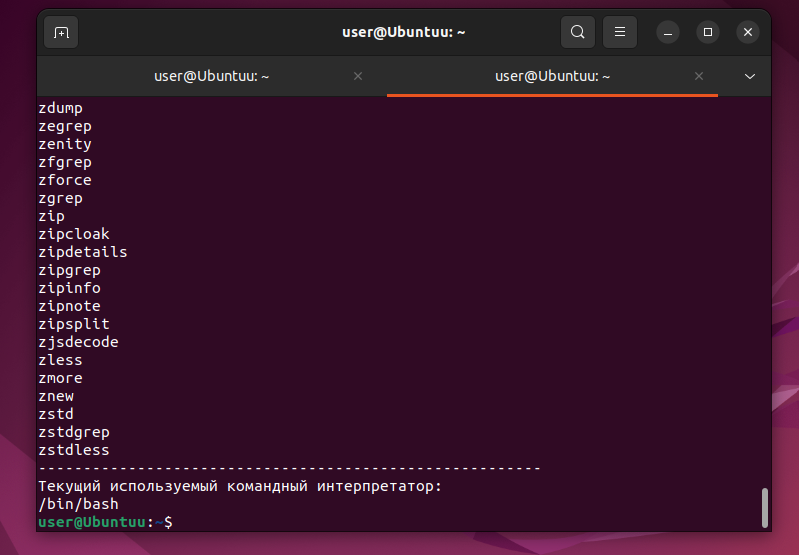
```

ls /bin

```

Эта команда покажет список файлов командных интерпретаторов, если они установлены на вашей системе. Однако учтите, что на некоторых системах они могут быть доступны в других каталогах, таких как `/usr/bin` или `/usr/local/bin`,

****

****

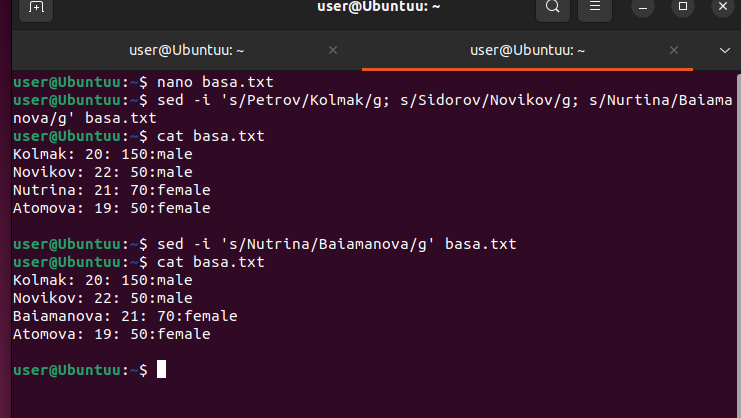
**Задание 2**

**6. Используйте текст первого примера, заменить некоторые фамилии на**

**фамилии членов вашей команды (например, Nutrina заменить на**

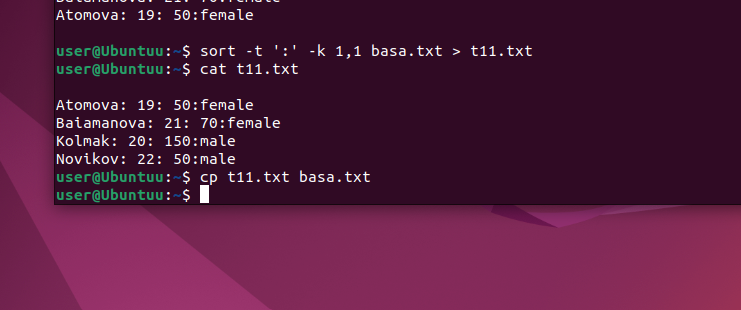
**Stankewich).**

Выполняем эту задачу в терминале с помощью команды sed, чтобы заменить фамилии в файле basa.txt

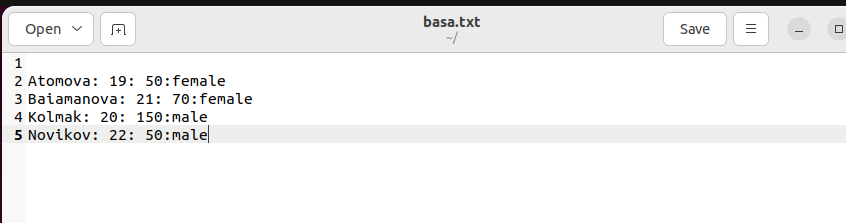
****

**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).**

* Создать отсортированный файл t11.txt, используя команду sort: Эта команда сортирует содержимое файла basa.txt по первому полю (фамилия) и сохраняет результат в файл t11.txt. Разделитель полей задан как : с помощью -t ':'.
* Проверить содержимое файла t11.txt, чтобы убедиться, что данные отсортированы.
* вернуть исходное имя файла basa.txt, вы можете скопировать содержимое t11.txt обратно в basa.txt: После выполнения этой команды файл basa.txt будет содержать отсортированные данные, а t11.txt можно удалить, если он больше не нужен.

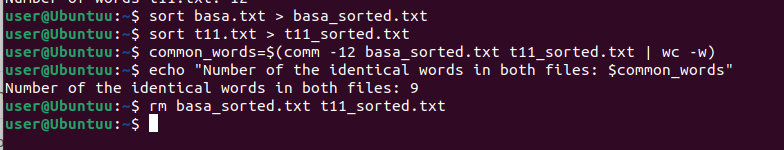






**8. Подсчитайте число одинаковых слов в обоих файлах. Для подсчета**

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

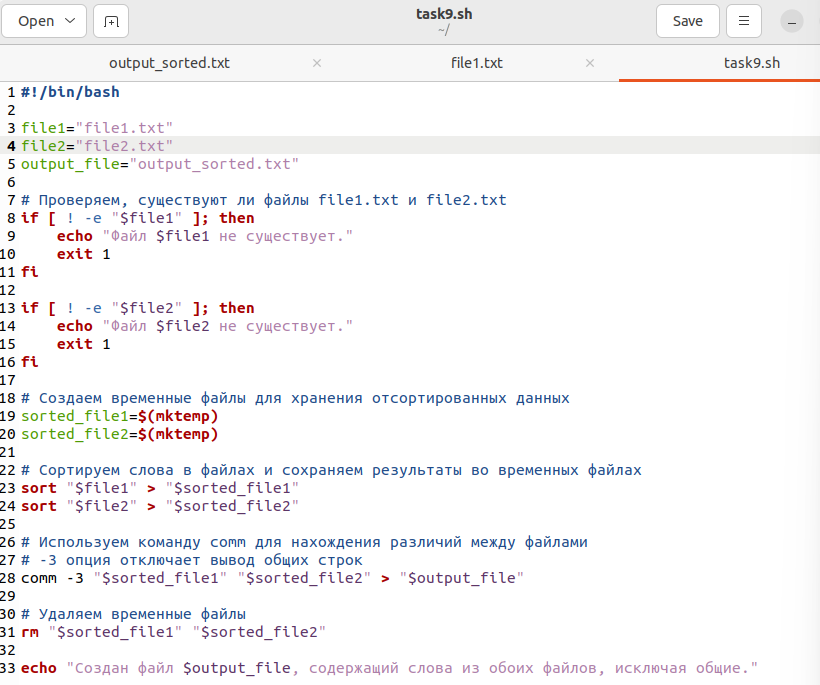


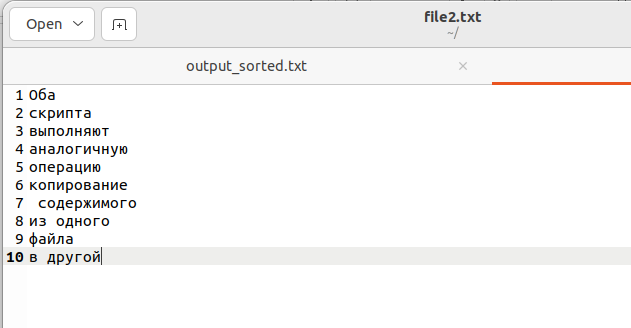
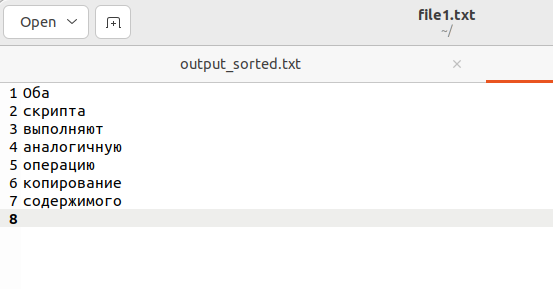
* подсчитать количество одинаковых слов в двух файлах и не только узнать количество слов в каждом файле, но и сравнить их, то можно воспользоваться командой comm

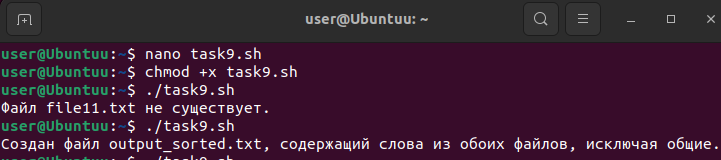
Этот скрипт сначала сортирует слова в файлах basa.txt и t11.txt, сохраняя результаты во временных файлах basa\_sorted.txt и t11\_sorted.txt. Затем он использует команду comm для сравнения отсортированных файлов и выводит слова, которые присутствуют в обоих файлах (-12 опции). Наконец, команда wc -w используется для подсчета числа слов в полученном списке общих слов, и результат выводится на экран.В конце # Удаляем временные файлы при помощи rm.

**9. Написать скрипт, который создает отсортированный файл, содержащий**

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

****

****

****

Этот скрипт выполняет следующие шаги:

Проверяет, существуют ли файлы file1.txt и file2.txt. Если один из файлов отсутствует, скрипт выдаст сообщение об ошибке и завершит выполнение.

Создает временные файлы sorted\_file1 и sorted\_file2 для хранения отсортированных данных из file1.txt и file2.txt.

Сортирует слова в файлах file1.txt и file2.txt и сохраняет отсортированные результаты во временных файлах sorted\_file1 и sorted\_file2

Использует команду comm с опцией -3 для нахождения различий между отсортированными файлами и сохраняет результат в output\_file.

Удаляет временные файлы.

Выводит сообщение о том, что создан файл output\_sorted.txt, содержащий слова из обоих файлов, исключая общие.

Теперь в файле output\_sorted.txt будут содержаться только уникальные слова из file1.txt и file2.txt.



**11.Вывести отсортированный список имен файлов, начальная часть имени**

**есть script.**



Эта команда выполняет следующие действия:

ls - выводит список файлов в текущем каталоге.

grep "^script" - фильтрует этот список, оставляя только строки, которые начинаются с "script" (символ '^' указывает на начало строки).

sort - сортирует отфильтрованный список по алфавиту.

**12.Написать скрипт, который в каждой строке файла оставляет только**

**буквенные символы, а остальные символы выбрасывает.**

Пример.

Пусть содержимое файла есть:

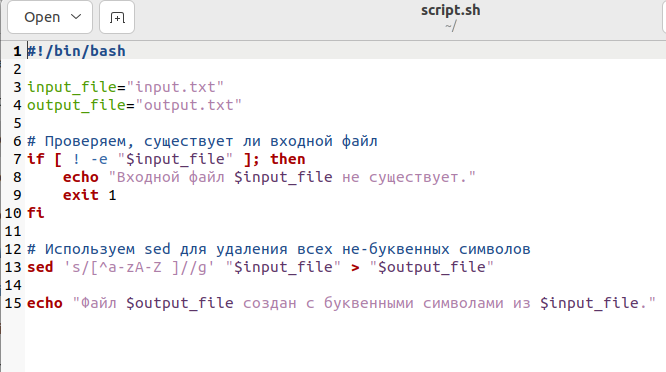
Privet year 2022

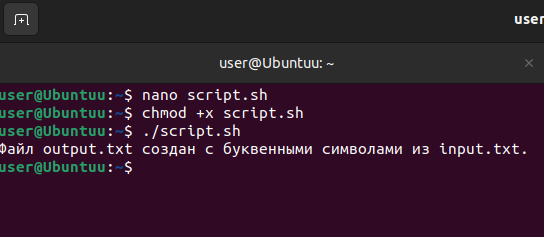
Good bye year 2021.

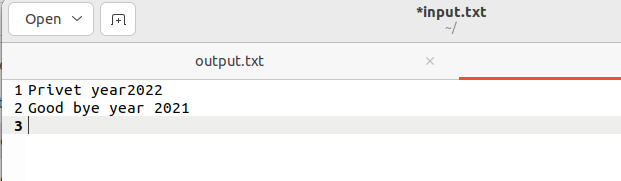
Должны получить такой файл:

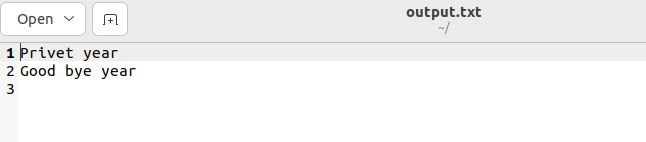
Privet year

Good bye year

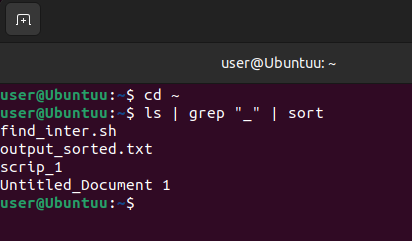






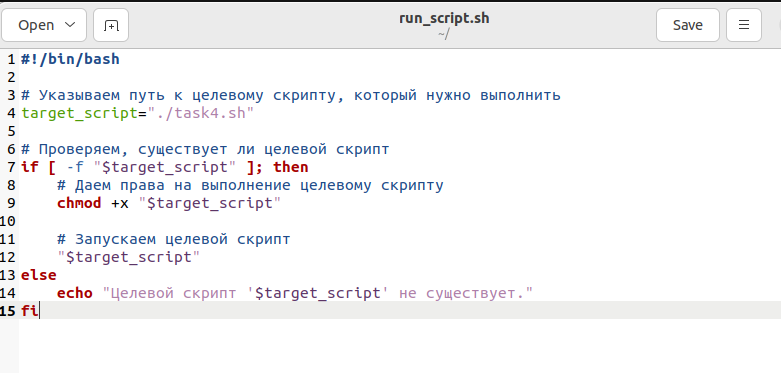


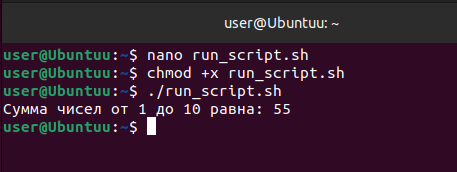
**13Вывести упорядоченный список имен файлов, в именах которых содержится символ подчеркивания, например, script\_sort.**

****

**Задание 3**

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

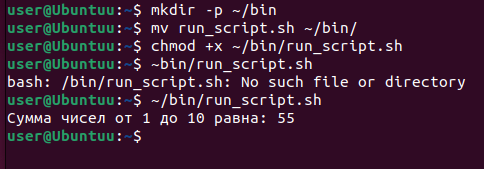




Этот скрипт сначала проверяет, существует ли целевой скрипт (your\_script.sh), и если он существует, то дает ему права на выполнение с помощью chmod +x и запускает его. Если целевой скрипт не существует, скрипт run\_script.sh выдаст сообщение об ошибке.

16.Расположите скрипт в одной из стантадртных директорий системы, и

запустите его из домашней директории пользователя.



Чтобы расположить скрипт в одной из стандартных директорий системы и запустить его из домашней директории пользователя, можно переместить скрипт в одну из директорий, которые входят в переменную PATH. Обычно, для пользовательских скриптов, вы можете выбрать директорию ~/bin (где ~ обозначает домашнюю директорию пользователя).

* Создайте директорию bin в вашей домашней директории, если она не существует:
* Переместите ваш скрипт run\_script.sh в эту директорию:
* Убедитесь, что ваш скрипт имеет права на выполнение:
* Теперь можно запустить скрипт из домашней директории пользователя, даже если находитесь в другой директории: Система автоматически будет искать исполняемые скрипты в директории ~/bin.

**Контрольные вопросы:**

**Скрипты или сценарии командной оболочки:**

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

Файлы скриптов должны иметь права на выполнение (execute permissions), чтобы их можно было запускать как исполняемые программы. В Linux и Unix системах, чтобы дать файлу права на выполнение, вы должны установить бит выполнения для соответствующего файла.

Права на выполнение устанавливаются с помощью команды `chmod`, и их можно установить следующими способами:

1. Через символическое представление:

- `chmod +x filename` - добавить право выполнения.

- `chmod -x filename` - удалить право выполнения.

- `chmod u+x filename` - добавить право выполнения для владельца.

- `chmod go+x filename` - добавить право выполнения для группы и остальных.

2. Через числовое представление:

- `chmod 755 filename` - устанавливает права 755 (rwxr-xr-x), что означает право выполнения для владельца, и чтение/выполнение для группы и остальных.

Обычно файлы скриптов, которые должны выполняться, имеют права на выполнение для владельца. Например:

```bash

chmod u+x script.sh

```

Эта команда дает право на выполнение только владельцу файла `script.sh`.

Убедитесь, что у скрипта есть права на выполнение, чтобы он мог быть запущен как исполняемая программа.

**Перенаправление ввода/вывода и потоки данных:**

**8) Что такое потоки данных, и какие стандартные потоки данных**

**существуют в системе? В какой системной директории располагаются**

**их файлы?**

Потоки данных (также известные как стандартные потоки) представляют собой механизм передачи информации между программами и операционной системой в Linux и других Unix-подобных системах. Они представляют собой абстракцию ввода/вывода и обеспечивают основной способ взаимодействия программ в текстовом режиме.

В Linux существуют три стандартных потока данных:

1. \*\*Стандартный ввод (stdin)\*\*: Поток данных, через который программа получает входные данные. Обычно это клавиатура, но может быть перенаправлен на чтение из файлов или других источников.

2. \*\*Стандартный вывод (stdout)\*\*: Поток данных, через который программа выводит результаты своей работы. Обычно это экран (терминал), но также может быть перенаправлен на запись в файлы или другие места.

3. \*\*Стандартный поток ошибок (stderr)\*\*: Поток данных, через который программа выводит сообщения об ошибках и диагностическую информацию. Как и stdout, stderr обычно направлен на экран, но может быть перенаправлен на запись в файлы или другие места.

Стандартные потоки данных в Linux реализованы как файлы в системе. Они обозначаются следующими файловыми дескрипторами:

- `stdin` - файловый дескриптор 0.

- `stdout` - файловый дескриптор 1.

- `stderr` - файловый дескриптор 2.

Их файлы находятся в системной директории `/dev/`, и они имеют следующие имена:

- `/dev/stdin` - файл, представляющий stdin (файловый дескриптор 0).

- `/dev/stdout` - файл, представляющий stdout (файловый дескриптор 1).

- `/dev/stderr` - файл, представляющий stderr (файловый дескриптор 2).

Эти файлы можно использовать для перенаправления потоков данных в программе. Например, чтобы перенаправить stdout в файл `output.txt`, вы можете использовать следующую команду:

```bash

command > output.txt

```

Потоки данных - это важная концепция в Linux и Unix-подобных системах, которая позволяет эффективно взаимодействовать с программами и манипулировать вводом и выводом.