

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14

дисциплина: Архитектура компьютера

Студентка: Симбине Камила Шеймиле

Группа: НПИбд-03-23

МОСКВА

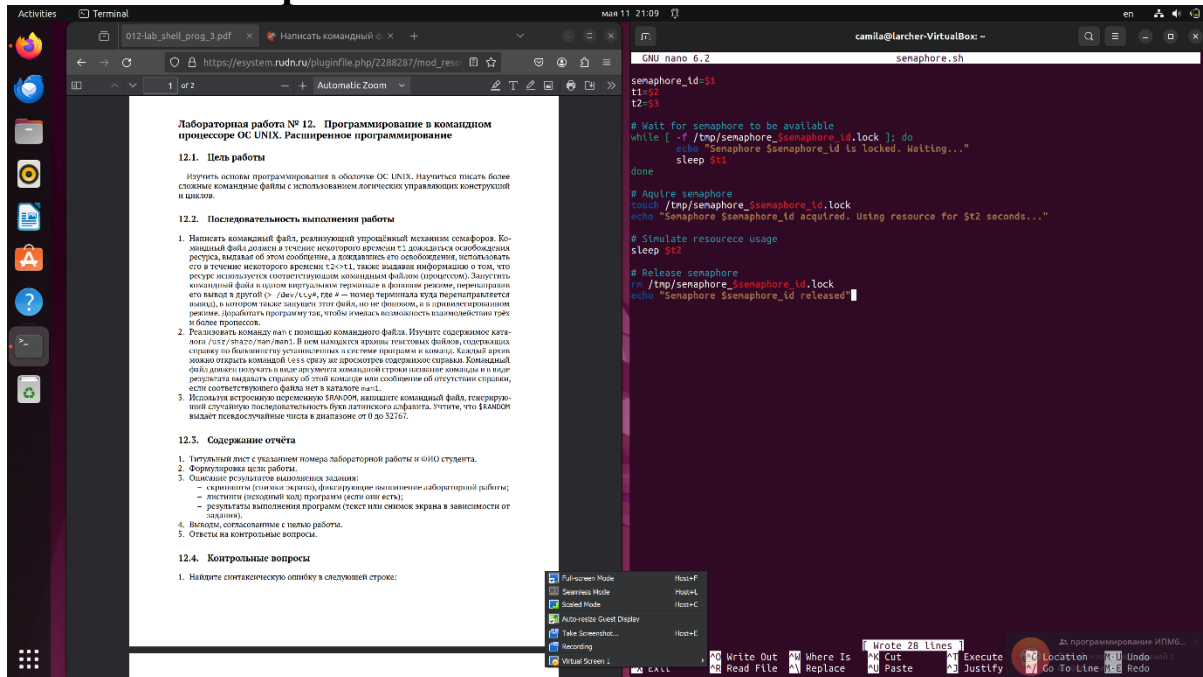
2024 г.

Расширенное программирование

Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Выполнение работы



Задача: Создать сценарий (semaphore.sh) для реализации упрощенного механизма семафоров, который позволяет процессам ожидать и использовать общий ресурс в течение определенного периода времени.

Объяснение сценария (semaphore.sh):

Ввод аргументов: Скрипт принимает три аргумента командной строки: <semaphore_id>, <t1> и <t2>.

<semaphore_id>: идентификатор семафора.

<t1>: Время ожидания (в секундах), если семафор в данный момент заблокирован.

<t2>: Время использования ресурса (в секундах) после получения семафора.

Логика семафора:

Скрипт проверяет наличие файла блокировки семафора (/tmp/semaphore_\$semaphore_id.lock).

Если файл блокировки существует, это означает, что семафор в данный момент используется. Скрипт отобразит сообщение, указывающее на то, что он ожидает (семафор <semaphore_id> заблокирован. Ожидание...) и будет находиться в режиме ожидания в течение <t1> секунд перед повторной проверкой.

Как только семафор становится доступным (файл блокировки не существует), скрипт создает файл блокировки (/tmp/semaphore_\$semaphore_id.lock), указывая, что он получил

семафор.

Затем он переходит к моделированию использования ресурсов, переходя в спящий режим на $\langle t2 \rangle$ секунд.

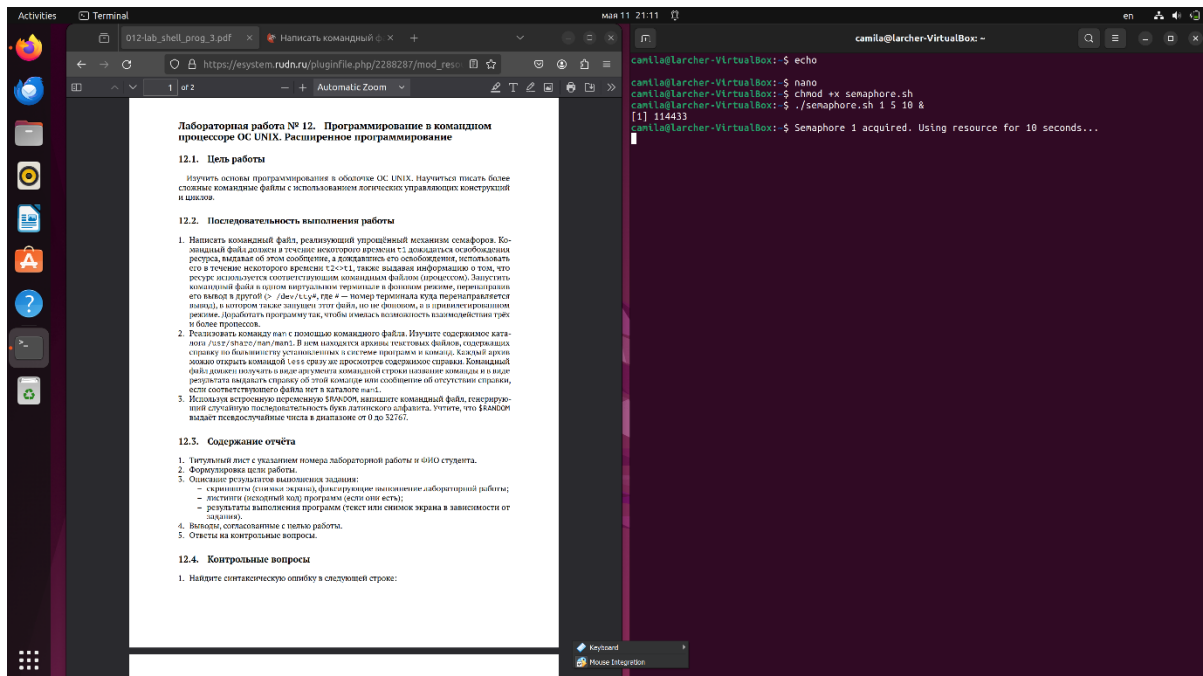
После завершения использования ресурсов скрипт разблокирует семафор, удалив файл блокировки.

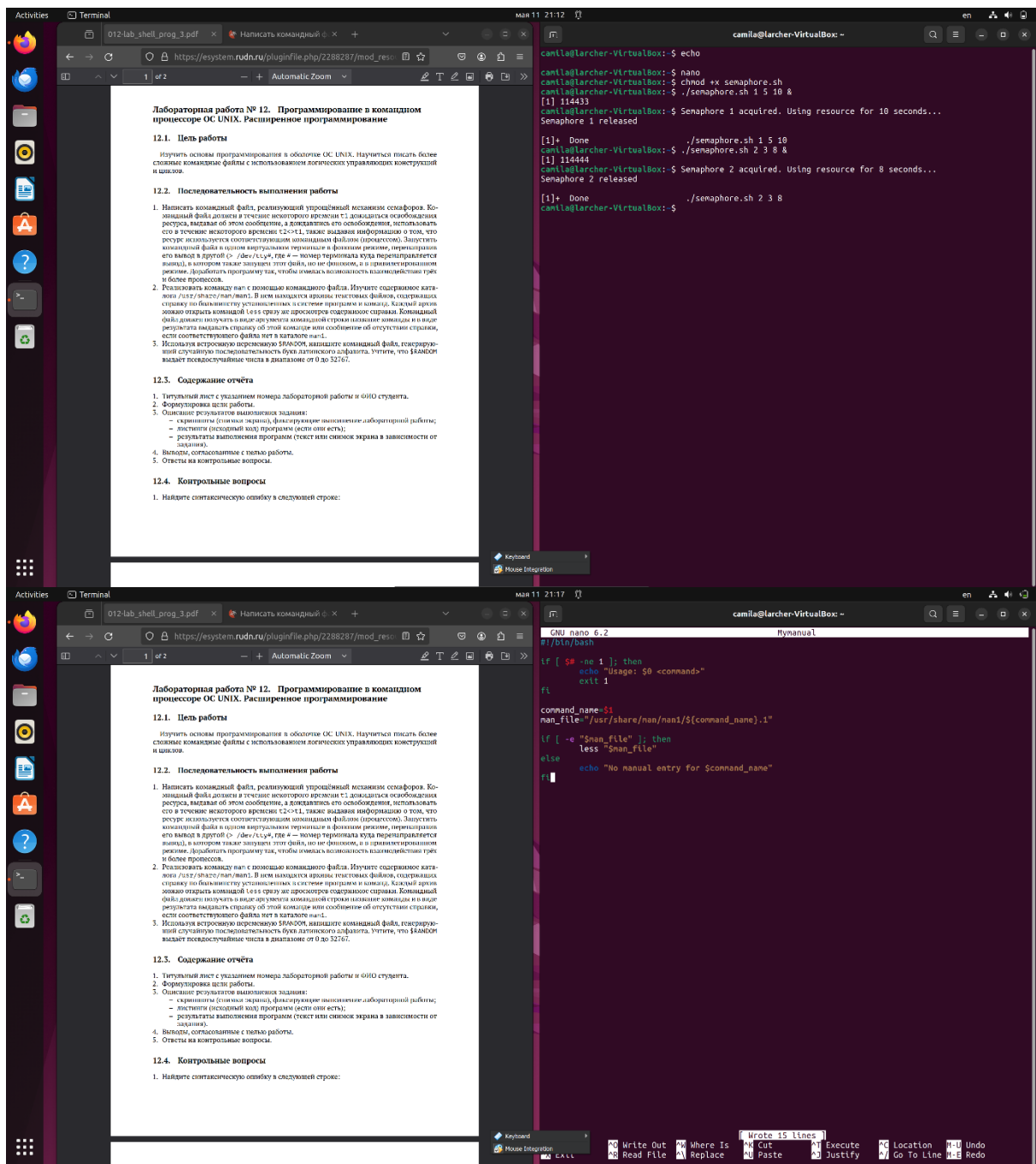
Ожидаемый результат:

Запуск нескольких экземпляров `semaphore.sh` с разными идентификаторами семафора (`<semaphore_id>`) продемонстрирует совместное использование ресурсов и синхронизацию.

Процессы будут ждать ($\langle t1 \rangle$ секунд), пока семафор станет доступен, если он используется в данный момент.

После получения доступа каждый процесс будет использовать общий ресурс в течение определенного времени ($\langle t2 \rangle$ секунд), прежде чем передать его для использования другим процессом.





Задача: Создать сценарий (myman.sh), который имитирует поведение команды man, но специально для команд со связанными страницами руководства (файлы *.1) в /usr/share/man/man1.

Пояснение к сценарию (myman.sh):

Входной аргумент: Скрипт ожидает, что в качестве аргумента будет указано одно имя команды (<команда>).

Поиск по странице вручную:

Он создает путь к соответствующему файлу подкачки вручную (/usr/share/man/man1/<команда>.1).

Если файл подкачки вручную существует (отметьте-f), скрипт использует меньше ресурсов для отображения его содержимого, что упрощает навигацию.

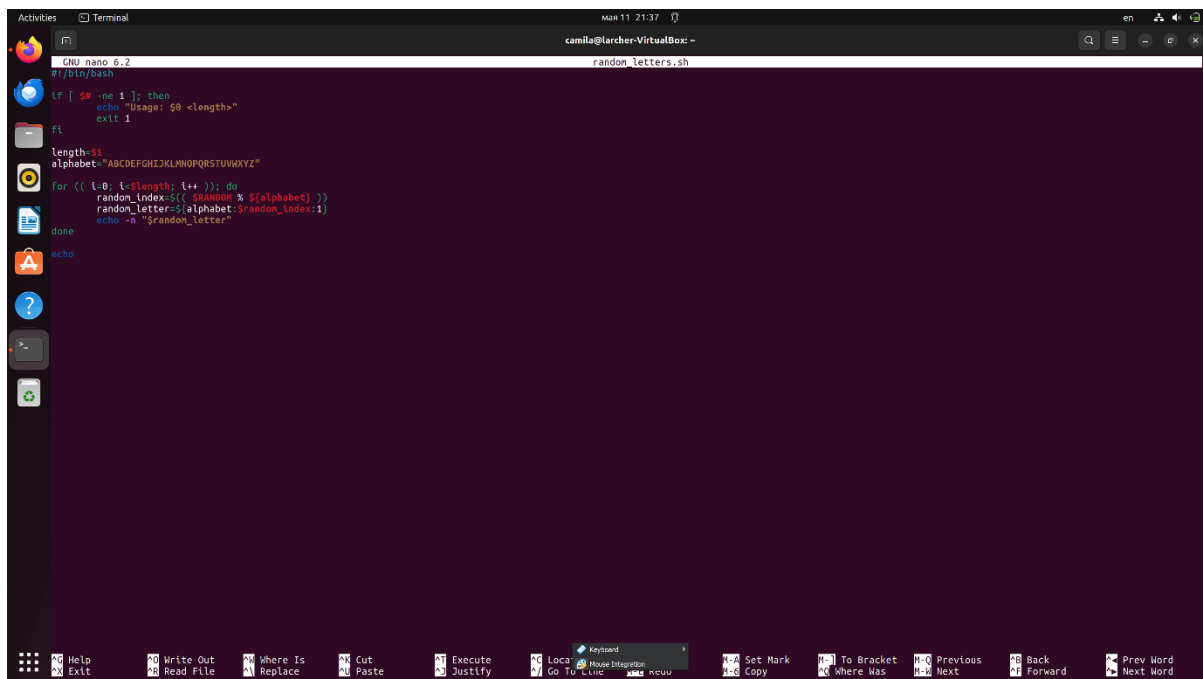
Если файл подкачки руководства не существует, он выводит сообщение о том, что для указанной команды не существует записи вручную.

Ожидаемый результат:

При запуске ./myman.sh <команда>:

Если для <команды> существует страница руководства, скрипт отобразит ее содержимое, используя less.

Если страница руководства не существует, она сообщит пользователю, что для <command> нет ручного ввода.



```
GNU nano 6.2
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Usage: $0 <length>"
    exit 1
fi

length=$1
alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"

for (( i=0; i<length; i++ )); do
    random_index=$((RANDOM % ${#alphabet}))
    random_letter=${alphabet:$random_index:1}
    echo -n "$random_letter"
done

echo
```

Задача: Создать скрипт (random_letters.sh), который генерирует случайную последовательность букв латинского алфавита.

Скрипт (random_letters.sh) Пояснение:

Входной аргумент: Скрипт ожидает один аргумент (<длина>), который определяет длину генерируемой случайной последовательности.

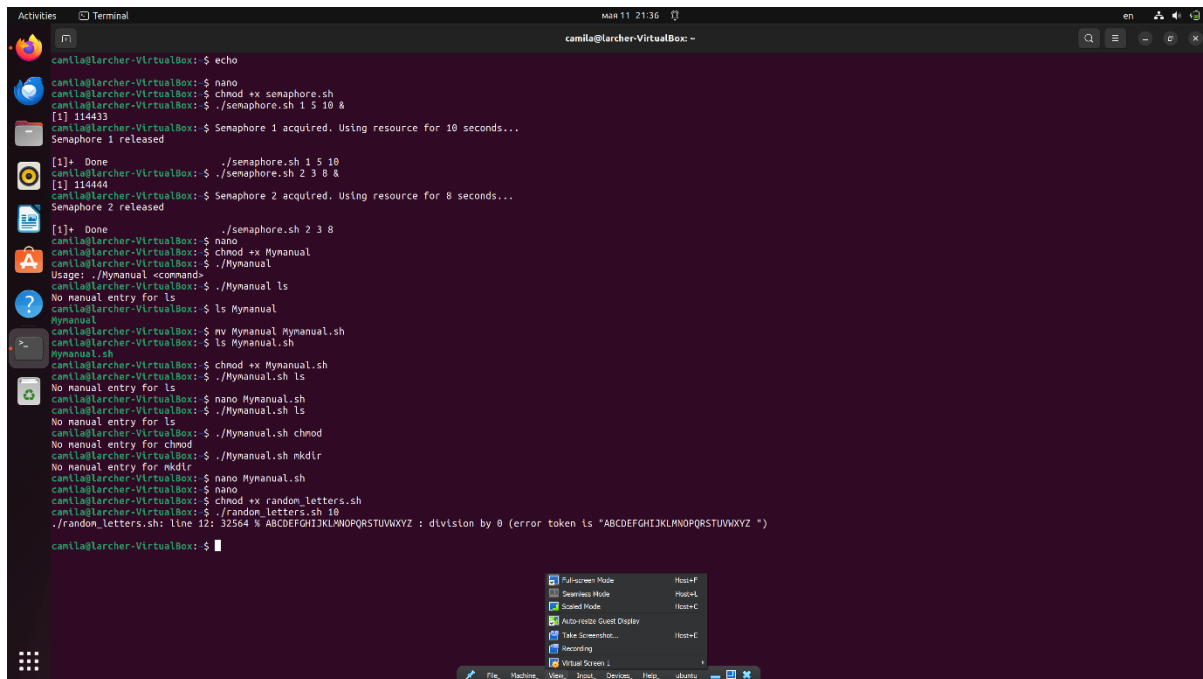
Генерация случайных букв:

Для генерации случайных букв <длина> используется цикл.

Для каждой итерации случайным образом выбирается буква из алфавита (в

верхнем регистре) с использованием переменной \$RANDOM.
Выбранные буквы объединяются в случайную последовательность.
Ожидаемый результат:

При запуске ./random_letters.sh <длина>:
Скрипт выведет случайную последовательность <длина> заглавных букв латинского алфавита.



```
canlla@larcher-VirtualBox:~$ echo
canlla@larcher-VirtualBox:~$ nano
canlla@larcher-VirtualBox:~$ chmod +x senaphore.sh
canlla@larcher-VirtualBox:~$ ./senaphore.sh 1 5 10 &
[1] 114433
canlla@larcher-VirtualBox:~$ Semaphore 1 acquired. Using resource for 10 seconds...
Semaphore 1 released
[1]+ Done ./senaphore.sh 1 5 10
canlla@larcher-VirtualBox:~$ ./senaphore.sh 2 3 8 &
[1] 114444
canlla@larcher-VirtualBox:~$ Semaphore 2 acquired. Using resource for 8 seconds...
Semaphore 2 released
[1]+ Done ./senaphore.sh 2 3 8
canlla@larcher-VirtualBox:~$ nano
canlla@larcher-VirtualBox:~$ chmod +x Mymanual
canlla@larcher-VirtualBox:~$ ./Mymanual
Usage: ./Mymanual <command>
canlla@larcher-VirtualBox:~$ ./Mymanual ls
No manual entry for ls
canlla@larcher-VirtualBox:~$ ls Mymanual
Mymanual
canlla@larcher-VirtualBox:~$ mv Mymanual Mymanual.sh
canlla@larcher-VirtualBox:~$ ls Mymanual.sh
Mymanual.sh
canlla@larcher-VirtualBox:~$ chmod +x Mymanual.sh
canlla@larcher-VirtualBox:~$ ./Mymanual.sh ls
No manual entry for ls
canlla@larcher-VirtualBox:~$ nano Mymanual.sh
canlla@larcher-VirtualBox:~$ ./Mymanual.sh ls
No manual entry for ls
canlla@larcher-VirtualBox:~$ ./Mymanual.sh chmod
No manual entry for chmod
canlla@larcher-VirtualBox:~$ ./Mymanual.sh mkdir
No manual entry for mkdir
canlla@larcher-VirtualBox:~$ nano Mymanual.sh
canlla@larcher-VirtualBox:~$ nano
canlla@larcher-VirtualBox:~$ chmod +x random_letters.sh
canlla@larcher-VirtualBox:~$ ./random_letters.sh 10
./random_letters.sh: line 12: 32564 % ABCDEFGHIJKLMNOPQRSTUVWXYZ : division by 0 (error token is "ABCDEFGHIJKLMNOPQRSTUVWXYZ ")
canlla@larcher-VirtualBox:~$
```

Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:

```
while [$1 != "exit"]
### Контрольные вопросы
```

1. ****Найдите синтаксическую ошибку в следующей строке:****

```
```bash
while [$1 != "exit"]
```
```

****Ответ:**** Синтаксическая ошибка заключается в том, что внутри условия `while` необходимо использовать пробелы вокруг операторов `[` и `]`. Правильный вариант:

```
```bash
```

```
while ["$1" != "exit"]
...
```

## 2. Как объединить (конкатенация) несколько строк в одну?

В Bash можно объединить строки с помощью оператора конкатенации `+` или просто написав строки подряд. Например:

```
```bash  
str1="Hello"  
str2="World"  
concatenated="${str1}${str2}" # конкатенация с  
использованием переменных  
concatenated="Hello""World"  # простое написание  
поряд  
...
```

3. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?

Утилита `seq` генерирует последовательность чисел. Её функционал можно реализовать в bash с помощью циклов или расширения фигурных скобок (brace expansion). Например:

```
```bash  
Через цикл for
for i in {1..10}; do
 echo "$i"
done

Через расширение фигурных скобок
echo {1..10}
...
```

#### **4. Какой результат даст вычисление выражения $\$(10/3)$ ?**

Результат вычисления выражения “ $\$(10/3)$ ” в `bash` будет “3”. Деление целых чисел в `bash` (с помощью оператора “ $(( ))$ ”) дает результат целочисленного деления.

#### **5. Укажите кратко основные отличия командной оболочки `zsh` от `bash`.**

- “`zsh`” поддерживает расширенный синтаксис и более продвинутые возможности по сравнению с “`bash`”.
- “`zsh`” имеет более интеллектуальное автодополнение и улучшенный интерфейс командной строки.
- “`zsh`” имеет некоторые различия в синтаксисе и поведении встроенных команд.

#### **6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`**

Данный синтаксис является корректным для объявления цикла “`for`” в `bash`. Он используется для итераций от начального значения “`a`” до значения “`LIMIT`”, увеличивая “`a`” на 1 с каждой итерацией.

#### **7. Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки?**

##### **Преимущества `bash`:**

- Простота и удобство в написании скриптов для автоматизации системных задач.
- Интеграция с UNIX-системами и мощными системными командами.
- Широкая поддержка на различных UNIX-подобных системах.

##### **Недостатки `bash`:**

- Ограниченные возможности для сложных алгоритмов и структур данных.



- Не подходит для разработки сложных приложений из-за ограниченной поддержки объектно-ориентированного программирования.

- Меньшая производительность по сравнению с компилируемыми языками.