# 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**

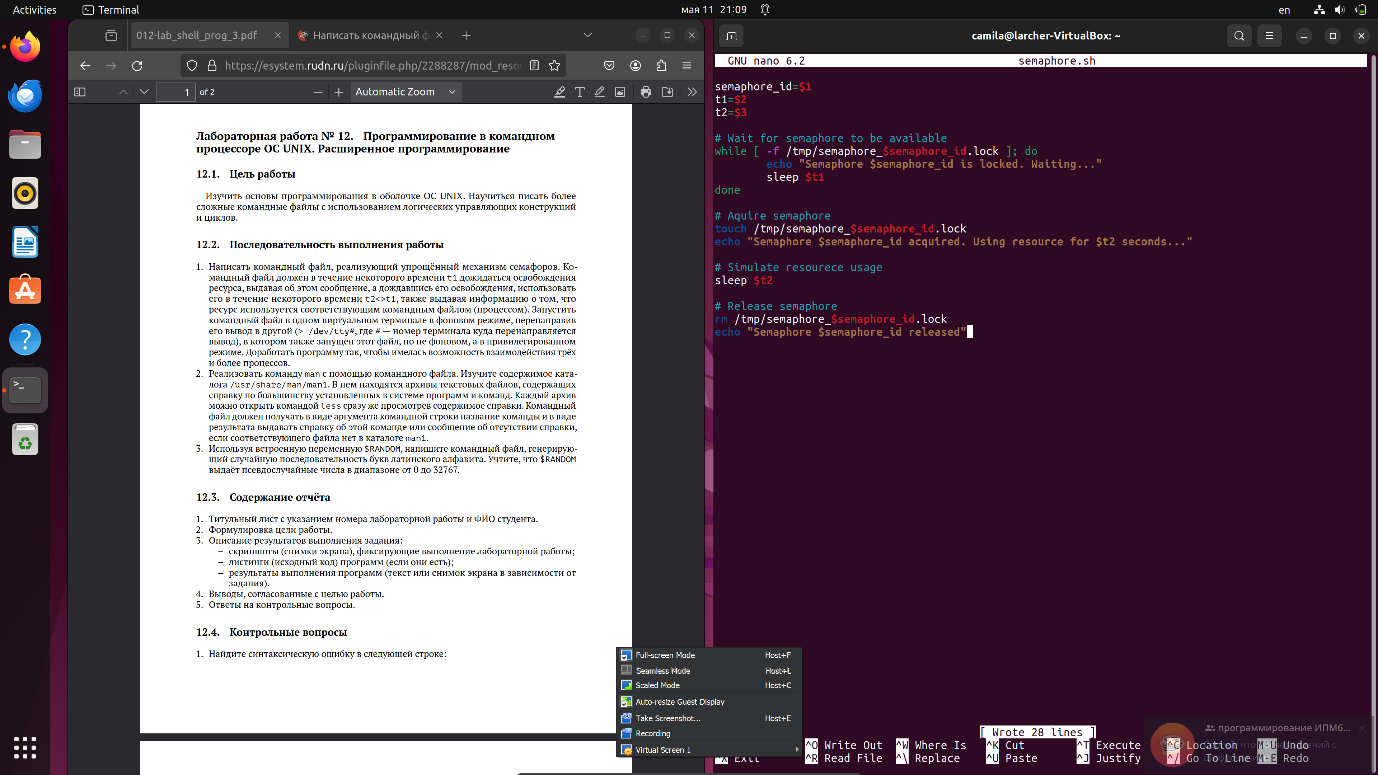
2024 г.

**Расширенное программирование**

**Цель работы**

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

**Ввыполнение 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 <t2> секунд.

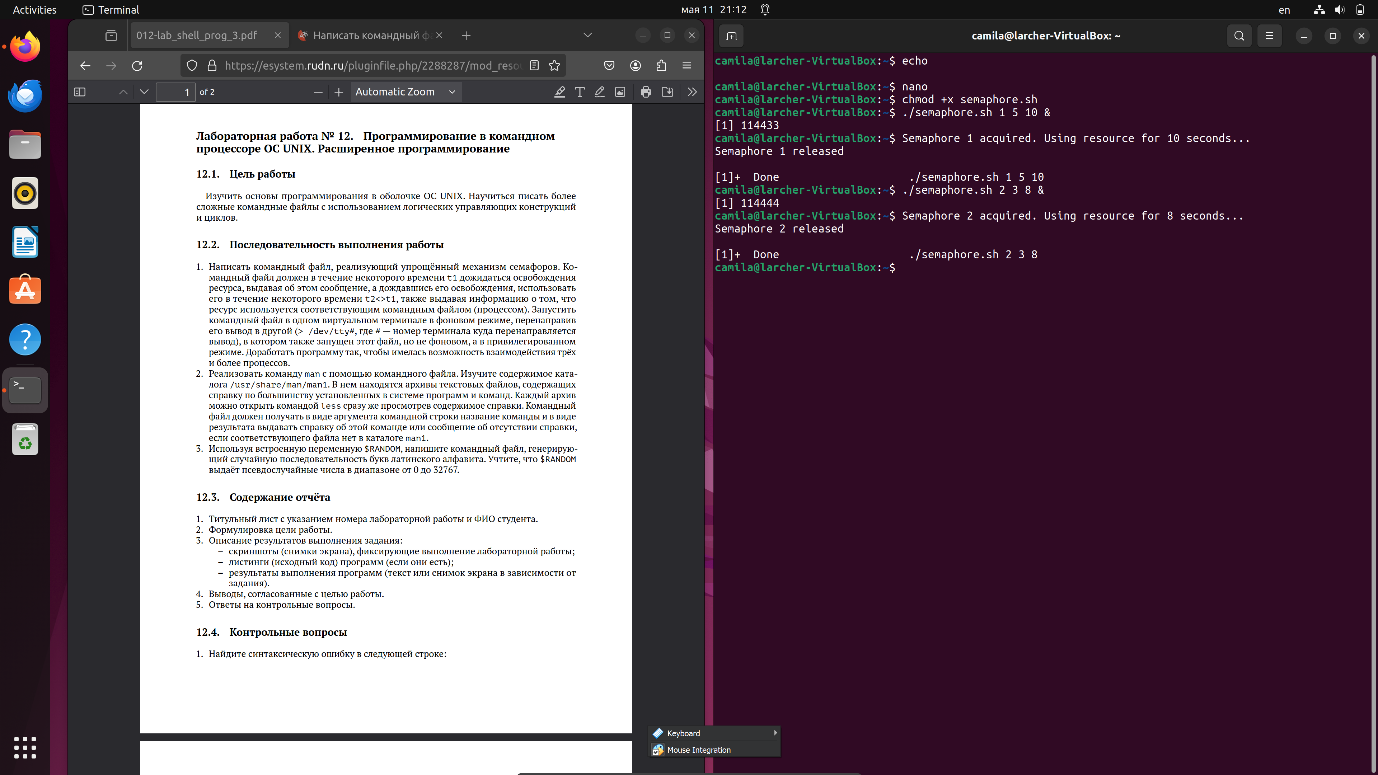
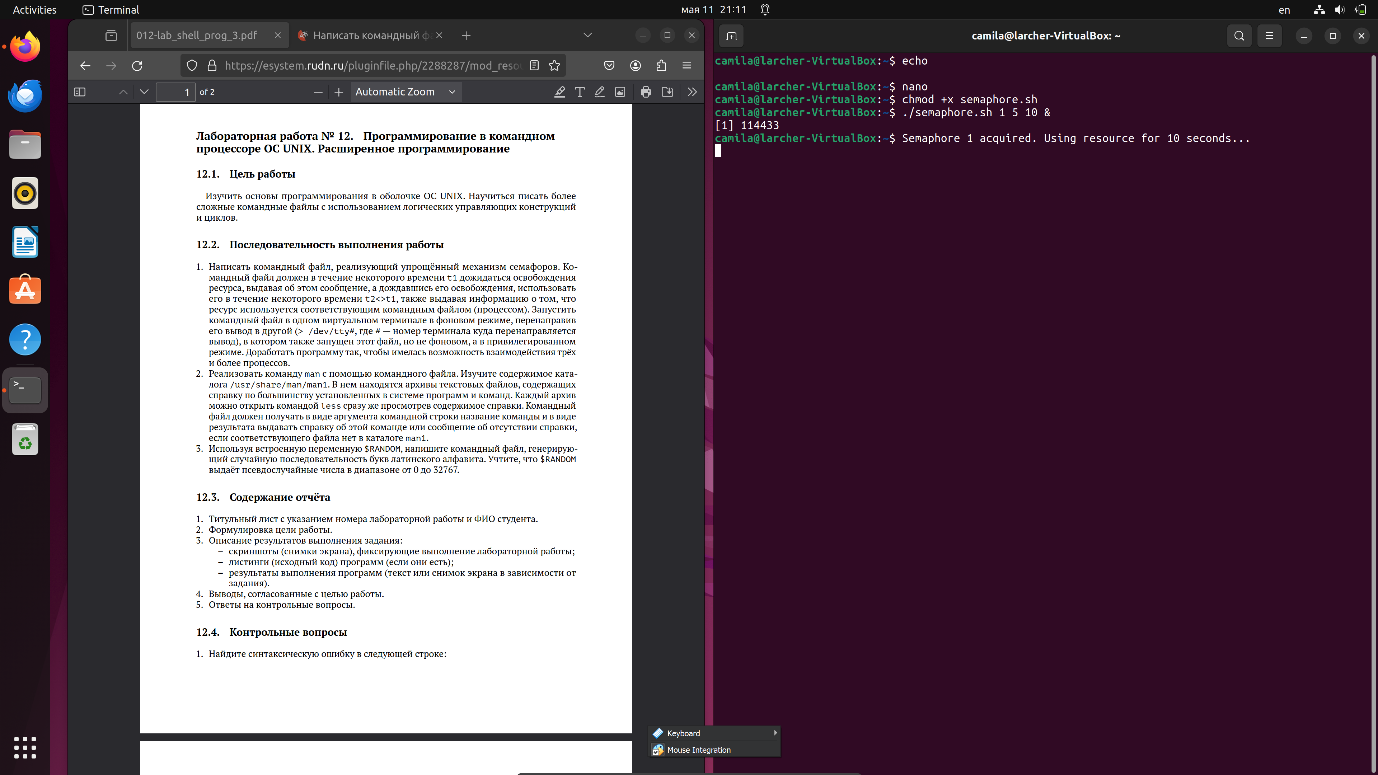
После завершения использования ресурсов скрипт разблокирует семафор, удалив файл блокировки.

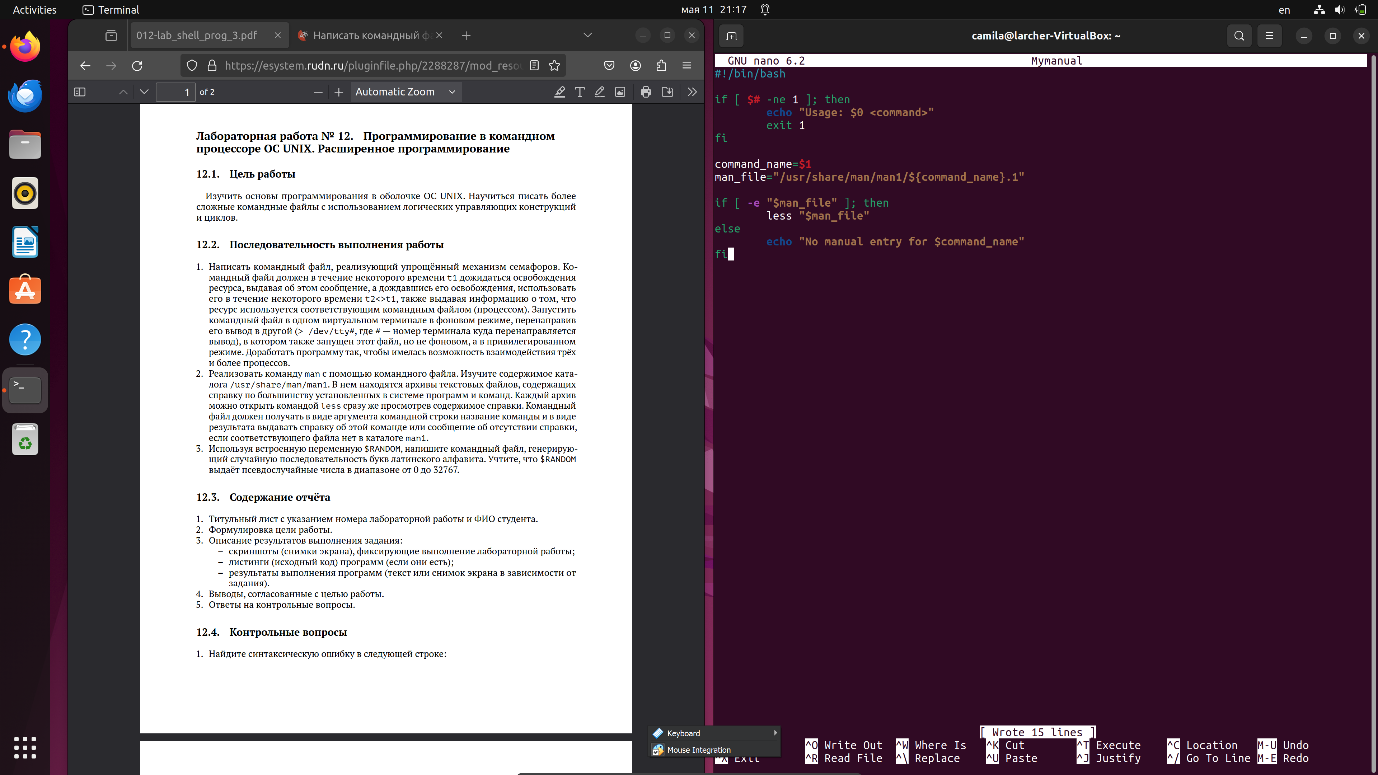
Ожидаемый результат:

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

Процессы будут ждать (<t1> секунд), пока семафор станет доступен, если он используется в данный момент.

После получения доступа каждый процесс будет использовать общий ресурс в течение определенного времени (<t2> 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.

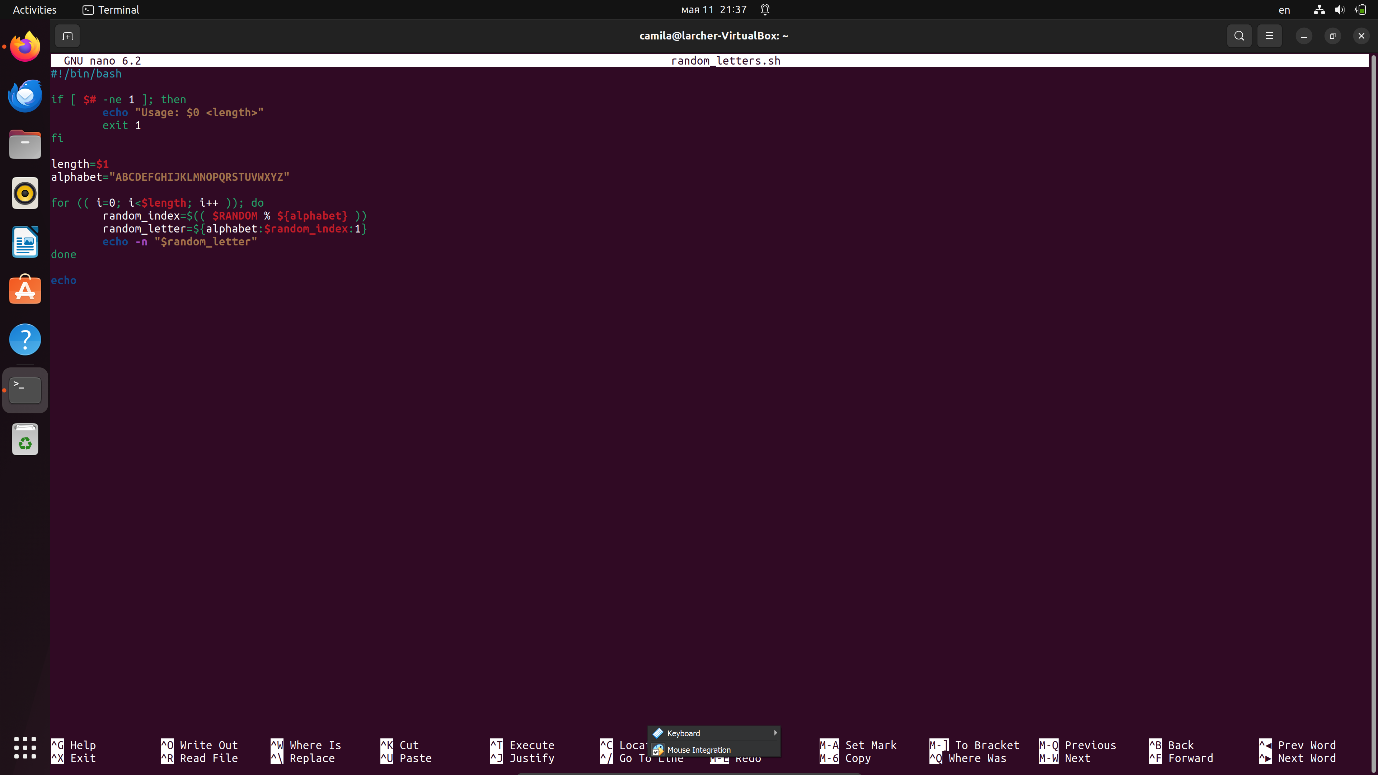
Если файл подкачки руководства не существует, он выводит сообщение о том, что для указанной команды не существует записи вручную.

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.

****

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.

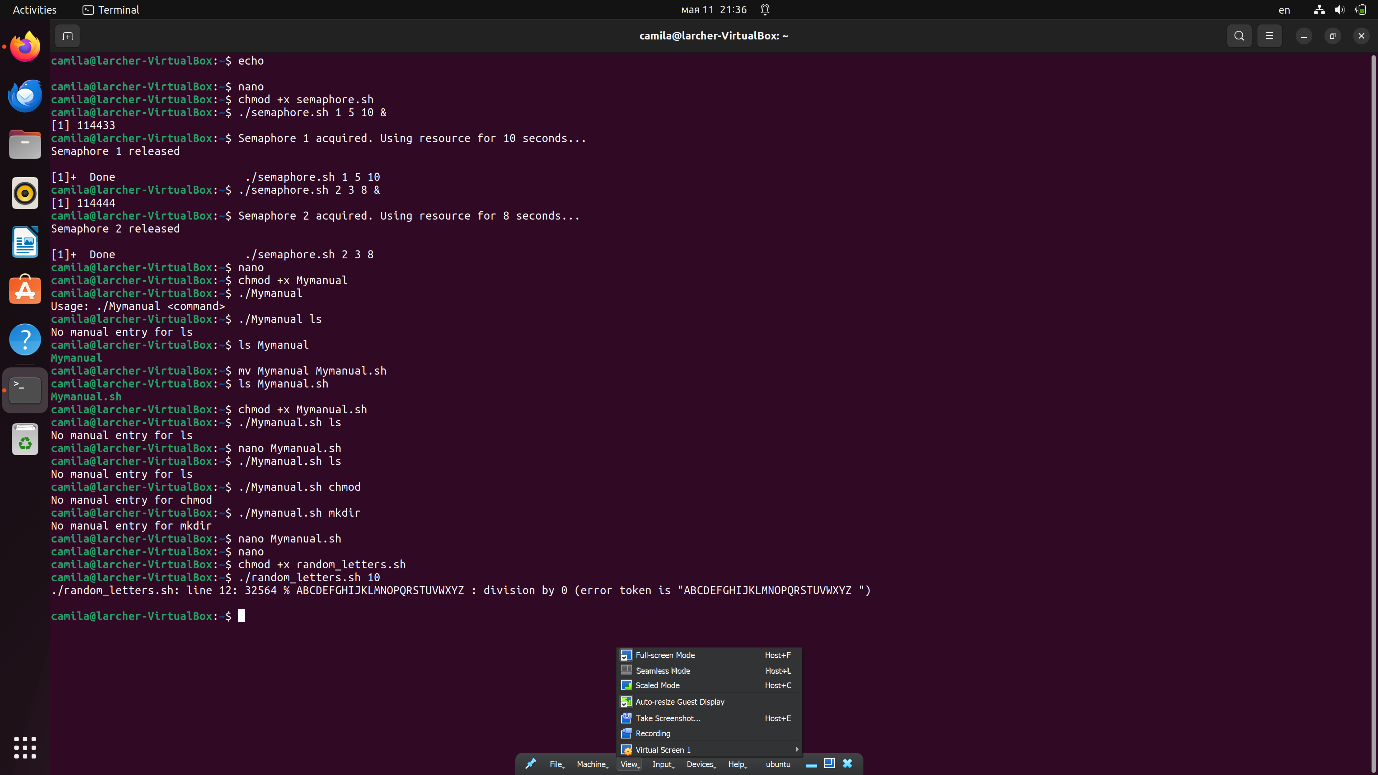
Для каждой итерации случайным образом выбирается буква из алфавита (в верхнем регистре) с использованием переменной $RANDOM.

Выбранные буквы объединяются в случайную последовательность.

Ожидаемый результат:

При запуске ./random\_letters.sh <длина>:

Скрипт выведет случайную последовательность <длина> заглавных букв латинского алфавита.

****

**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 подряд

```

**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.