Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №10
дисциплины «Анализ данных»

Выполнила:
Кубанова Ксения Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: синхронизация потоков

Цель: приобрести навыки работы с синхронизацией потоков

Порядок выполнения работы

Индивидуальное задание.

Для своего индивидуального задания лабораторной работы 2.23 необходимо организовать конвейер, в котором сначала в отдельном потоке вычисляется значение первой функции, после чего результаты вычисления должны передаваться второй функции, вычисляемой в отдельном потоке. Потоки для вычисления значений двух функций должны запускаться одновременно.

```
result_queue = queue.Queue()

i = 1
while True:
    # Создаем поток для вычисления части ряда
    part_thread = threading.Thread(target=calc_part, args=(x, i, result_queue))
    part_thread.start()

    # Получаем результат вычисления
    chis, znam = result_queue.get()
    part_thread.join()
```

Рисунок 1 – создание очереди

```
def calc_part(x, n, result_queue):
    chis = calc_chis(x, n)
    znam = calc_znam(n)
    result_queue.put((chis, znam))
```

Рисунок 2 – передача значение в очередь

Для выполнения поставленной задачи была использована очередь queue.Queue для передачи данных между потоками, которая действует следующим образом:

- Создается очередь result_queue для передачи результатов вычислений между потоками. (рис. 1)
- Функция calc_part вычисляет часть ряда (числитель и знаменатель) и помещает результаты в очередь. (рис. 2)

- Основной поток получает результаты из очереди и использует их для вычисления следующего элемента ряда.
- После проверки условия остановки основной поток завершает выполнение.

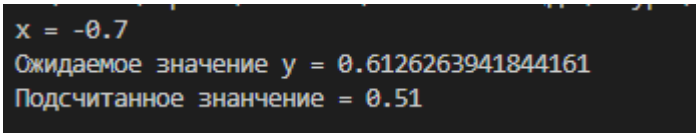

Рисунок 3 – результат выполнения программы

Итоговый код находится в ind.py.

Контрольные вопросы

1 Каково назначение и каковы приемы работы с Lock-объектом.

Lock (блокировка) используется для синхронизации доступа к общим ресурсам в многопоточной среде.

Основное назначение Lock - обеспечение монопольного доступа к общему ресурсу: только один поток может захватить блокировку (lock), остальные потоки ждут ее освобождения.

При работе с Lock-объектом используются методы acquire() для захвата блокировки и release() для ее освобождения.

2 В чем отличие работы с RLock-объектом от работы с Lock-объектом.

RLock (рекурсивная блокировка) является расширением обычной блокировки Lock и позволяет одному потоку захватывать блокировку несколько раз.

При использовании RLock можно захватывать блокировку множество раз в одном потоке без возникновения блокировки (deadlock).

В отличие от Lock, при работе с RLock поток может освобождать блокировку столько раз, сколько он ее захватил.

3 Как выглядит порядок работы с условными переменными?

Условные переменные используются для ожидания определенного условия и уведомления других потоков о его изменении.

Для работы с условными переменными в Python используется модуль threading и класс Condition.

Порядок работы с условными переменными включает захват блокировки, ожидание условия с помощью метода wait(), уведомление о изменении условия с помощью метода notify(), и освобождение блокировки.

4 Какие методы доступны у объектов условных переменных?

Основные методы доступные у объектов условных переменных в Python:

wait(timeout=None): ожидание изменения условия.

notify(n=1): уведомление одного или нескольких потоков о изменении условия.

notify_all(): уведомление всех потоков о изменении условия.

5 Каково назначение и порядок работы с примитивом синхронизации “семафор”?

Семафор - это примитив синхронизации, который ограничивает доступ к общему ресурсу определенным количеством потоков.

При создании семафора указывается количество доступных разрешений (permits).

Поток, желающий получить доступ к ресурсу, должен сначала захватить семафор методом acquire(), а затем освободить его методом release().

6 Каково назначение и порядок работы с примитивом синхронизации “событие”?

Событие - это примитив синхронизации, который используется для уведомления одного или нескольких потоков о наступлении определенного события.

Событие имеет два состояния: установлено (set) и сброшено (clear).

Потоки могут ожидать установки события с помощью метода wait(), а основной поток может установить или сбросить событие методами set() и clear().

7 Каково назначение и порядок работы с примитивом синхронизации “таймер”?

Таймер - это примитив синхронизации, который используется для запуска функции или метода по истечении определенного времени.

Порядок работы с таймером включает создание объекта таймера, указание времени задержки, запуск таймера методом start() и выполнение определенной функции по истечении времени.

8 Каково назначение и порядок работы с примитивом синхронизации “барьер”?

Барьер - это примитив синхронизации, который используется для синхронизации выполнения нескольких потоков.

Барьер имеет заданное количество потоков, которые должны достигнуть его, прежде чем все они будут разблокированы.

Потоки ожидают достижения барьера с помощью метода wait(), и когда все потоки достигают барьера, он разблокируется, и все потоки продолжают выполнение.

9 Сделайте общий вывод о применении тех или иных примитивов синхронизации в зависимости от решаемой задачи.

Выбор примитива синхронизации зависит от требований конкретной задачи.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки работы с синхронизацией потоков