МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

ФАИС

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
по дисциплине «**Операционные системы и среды**»

на тему: «**Синхронизация процессов**»

Выполнил: студент гр. ИП-32
Суховенко Э. С.
Принял: преподаватель
Процкая М. А.

Дата сдачи отчета: _____
Дата допуска к защите: _____
Дата защиты: _____

Гомель 2022

Цель: изучить типовые механизмы синхронизации процессов.

Задание

1. Разработать многопоточное приложение с использованием минимум двух потоков и различных средств синхронизации.

Таблица 1 Варианты заданий

№ варианта	Разделяемый ресурс	Механизм синхронизации
12	Два потока записывают и читают информацию из одного файла	мьютексы

2. Время входа в критическую секцию для каждого потока генерировать случайным образом.
3. В процессе работы приложения в консоль должна выводится информация о состоянии потока (работа в некритической секции, время входа и выхода из критической секции).
4. Убедиться в результативности применения синхронизации потоков, сравнив результаты работы программ с использованием и без использования средств синхронизации.

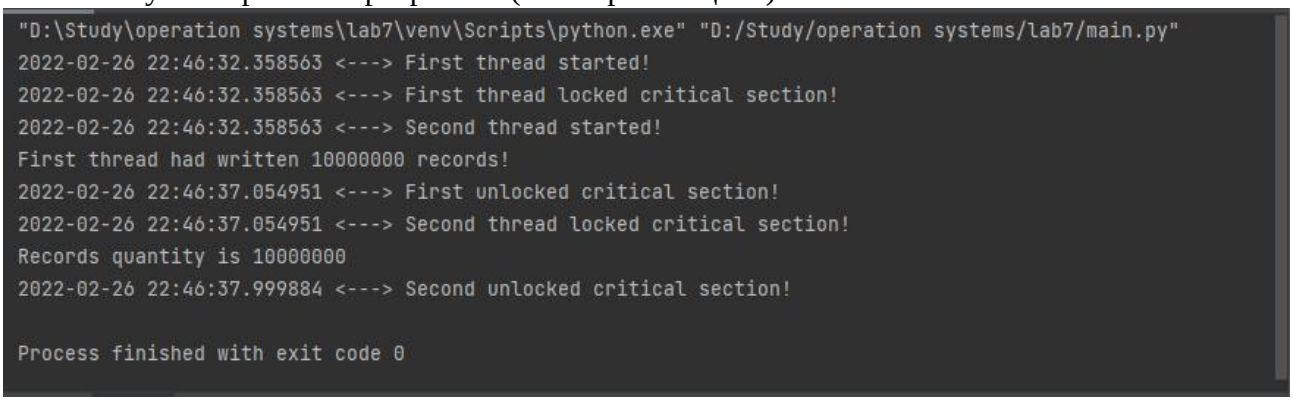
Выполнение

Код программы на ЯП Python:

```
import threading
import datetime

lock = threading.RLock()
quantity_records = 10000000


def first_thread_func(isLockUsed: bool):
    print(f'{datetime.datetime.now()} <---> First thread started!')
    if isLockUsed:
        lock.acquire()
        print(f'{datetime.datetime.now()} <---> First thread locked critical section!')
    try:
        with open('hallow.txt', 'w') as file:
            for i in range(quantity_records):
                file.write('first\n')
        print(f'First thread had written {quantity_records} records!')
    finally:
        if isLockUsed:
            lock.release()
            print(f'{datetime.datetime.now()} <---> First unlocked critical section!')


def second_thread_func(isLockUsed: bool):
    print(f'{datetime.datetime.now()} <---> Second thread started!')
    if isLockUsed:
        lock.acquire()
        print(f'{datetime.datetime.now()} <---> Second thread locked critical section!')
    try:
        with open('hallow.txt', 'r') as file:
            print(f'Records quantity is {sum(1 for _ in file)}')
    finally:
        if isLockUsed:
            lock.release()
```

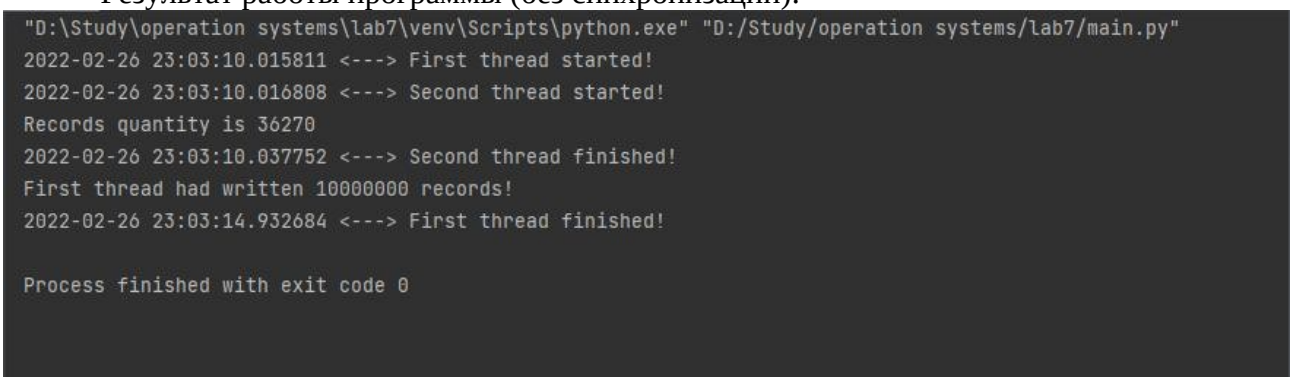
```
        print(f'{datetime.datetime.now()} <---> Second unlocked critical section!')


def main():
    with open('hallow.txt', 'w') as file:
        pass

    isLockUsed = False

    thread1 = threading.Thread(target=first_thread_func, args=([isLockUsed]))
    thread2 = threading.Thread(target=second_thread_func, args=([isLockUsed]))

    thread1.start()
    thread2.start()
    thread2.join()
    thread1.join()


if __name__ == "__main__":
    main()
```

Результат работы программы (с синхронизацией):



```
"D:\Study\operation systems\lab7\venv\Scripts\python.exe" "D:/Study/operation systems/lab7/main.py"
2022-02-26 22:46:32.358563 <---> First thread started!
2022-02-26 22:46:32.358563 <---> First thread locked critical section!
2022-02-26 22:46:32.358563 <---> Second thread started!
First thread had written 10000000 records!
2022-02-26 22:46:37.054951 <---> First unlocked critical section!
2022-02-26 22:46:37.054951 <---> Second thread locked critical section!
Records quantity is 10000000
2022-02-26 22:46:37.999884 <---> Second unlocked critical section!

Process finished with exit code 0
```

Результат работы программы (без синхронизации):



```
"D:\Study\operation systems\lab7\venv\Scripts\python.exe" "D:/Study/operation systems/lab7/main.py"
2022-02-26 23:03:10.015811 <---> First thread started!
2022-02-26 23:03:10.016808 <---> Second thread started!
Records quantity is 36270
2022-02-26 23:03:10.037752 <---> Second thread finished!
First thread had written 10000000 records!
2022-02-26 23:03:14.932684 <---> First thread finished!

Process finished with exit code 0
```

Вывод: были изучены типовые механизмы синхронизации процессов.