**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

Прокопенко А. Р.

Принял: преподаватель

Процкая М. А.

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

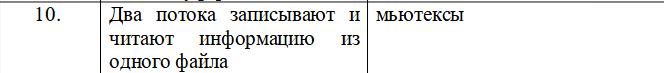
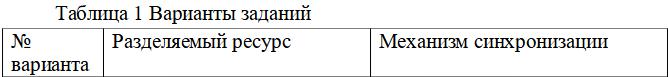
Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2022

**Цель**: изучить типовые механизмы синхронизации процессов.

**Задание**

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



1. Время входа в критическую секцию для каждого потока генерировать случайным

образом.

1. В процессе работы приложения в консоль должна выводится информация о состоянии потока (работа в некритической секции, время входа и выхода из критической секции).
2. 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():

print(f'{datetime.datetime.now()} <---> First thread started!')

lock.acquire()

print(f'{datetime.datetime.now()} <---> First thread locked critical section!')

try:

with open('hallow', 'w') as file:

for i in range(quantity\_records):

file.write('first\n')

print(f'First thread had written {quantity\_records} records!')

finally:

lock.release()

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

def second\_thread\_func():

print(f'{datetime.datetime.now()} <---> Second thread started!')

lock.acquire()

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

try:

with open('hallow', 'r') as file:

print(f'Records quantity is {sum(1 for \_ in file)}')

finally:

lock.release()

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

def main():

with open('hallow', 'w') as file:

pass

thread1 = threading.Thread(target=first\_thread\_func, args=())

thread2 = threading.Thread(target=second\_thread\_func, args=())

thread1.start()

thread2.start()

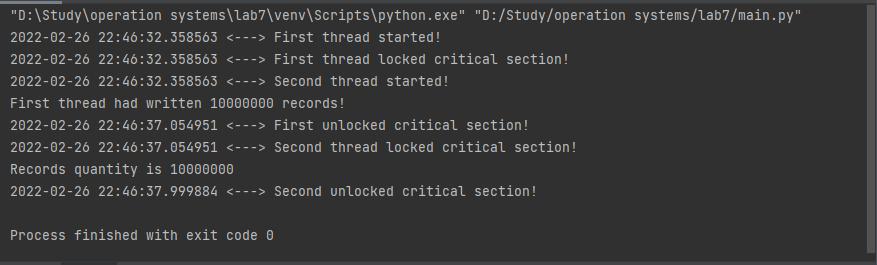
thread1.join()

thread2.join()

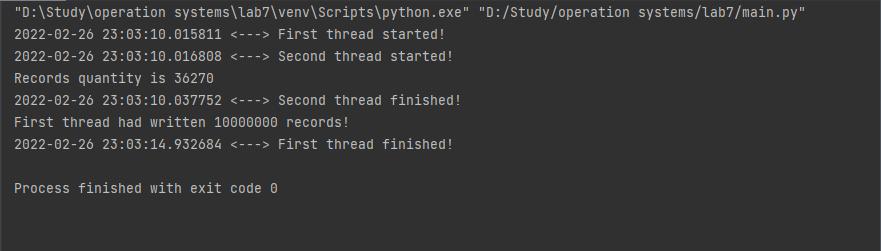
if \_\_name\_\_ == "\_\_main\_\_":

main()

Результат работы программы (с синхронизацией):



Результат работы программы (без синхронизации):



**Вывод**: были изучены типовые механизмы синхронизации процессов.