# 操作系统第一次作业报告

姓名：于奕骅 学号：120L021602 班级：2003202

## 作业要求

设计一个至少包含三个线程的应用，需要使用全局共享数据、线程间通信、线程同步技术。代码不少于50行，并撰写报告（包含源代码、效果截图、功能设计说明、操作系统进程、线程和IPC知识理解和运用）。

## 源代码

import threading

import queue

import random

# 全局共享数据：存储随机生成的数字（队列）

numbers = queue.Queue()

# 线程同步工具

mutex = threading.Lock()

semaphore = threading.Semaphore(0)

# 生产者线程

class ProducerThread(threading.Thread):

def \_\_init\_\_(self):

threading.Thread.\_\_init\_\_(self)

def run(self):

global numbers

while True:

# 随机生成数字并存储到队列中

number = random.randint(1, 10)

mutex.acquire()

# 同步访问全局队列

numbers.put(number)

print("Producer: produced %s" % number)

mutex.release()

semaphore.release()

# 通知消费者线程可以读取数字了

# 消费者线程

class ConsumerThread(threading.Thread):

def \_\_init\_\_(self):

threading.Thread.\_\_init\_\_(self)

def run(self):

global numbers

total = 0

while True:

semaphore.acquire()

# 等待生产者线程生成数字

mutex.acquire()

# 同步访问全局队列

if numbers.empty():

mutex.release()

break

number = numbers.get()

total += number

# 统计所有数字的和

print("Consumer: consumed %s" % number)

mutex.release()

print("Consumer: total is %s" % total)

# 主线程

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

producer\_thread = ProducerThread()

consumer\_thread\_1 = ConsumerThread()

consumer\_thread\_2 = ConsumerThread()

consumer\_thread\_3 = ConsumerThread()

producer\_thread.start()

consumer\_thread\_1.start()

consumer\_thread\_2.start()

consumer\_thread\_3.start()

producer\_thread.join()

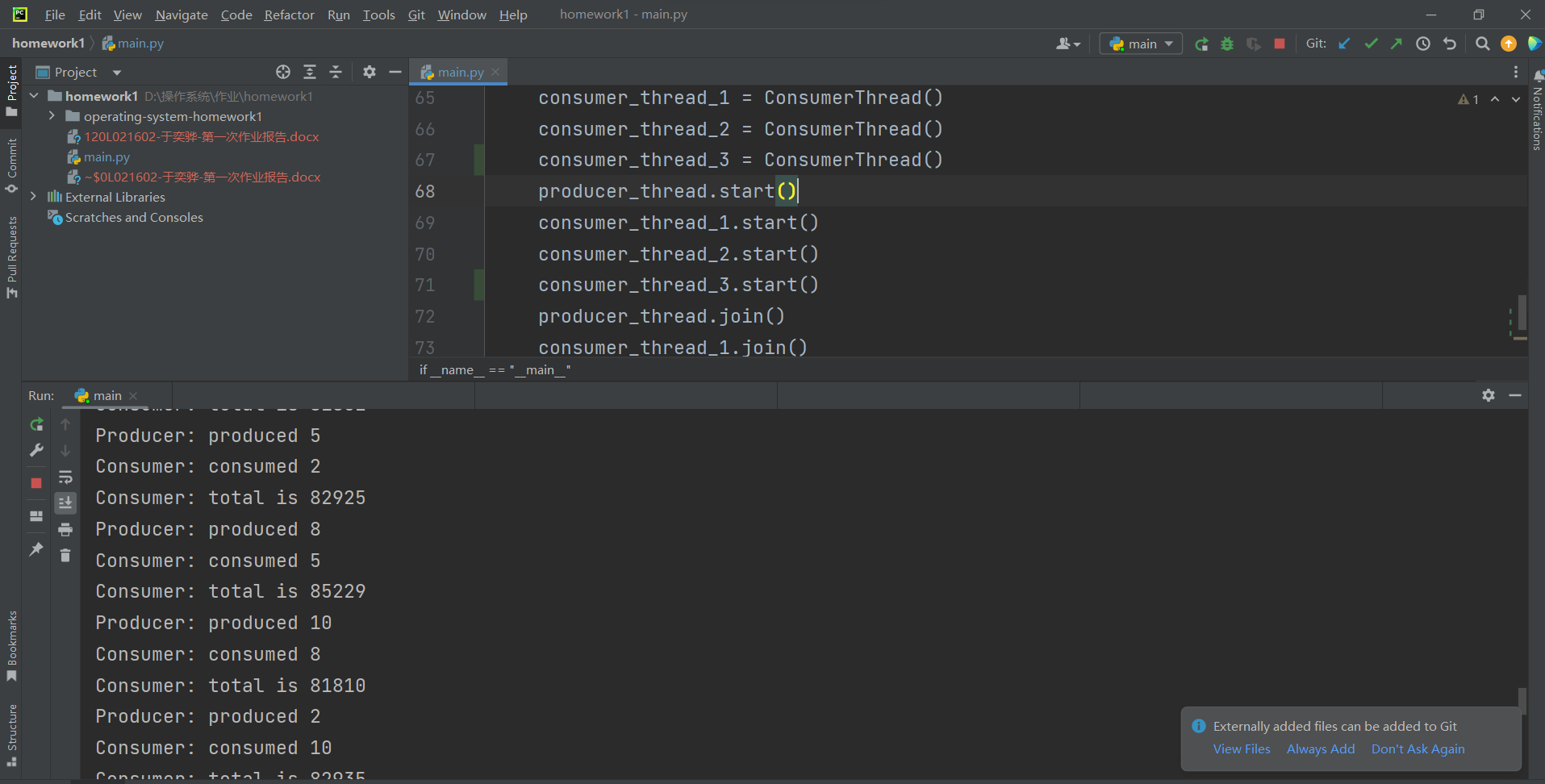
consumer\_thread\_1.join()

consumer\_thread\_2.join()

consumer\_thread\_3.join()

print("End of main thread.")

## 效果截图



## 功能设计说明

这是一个包含三个线程的应用，实现了全局数据共享、线程间通信、线程同步的功能。

该应用模拟了一个生产者-消费者模型，其中生产者线程随机生成数字并将其存储在全局队列中，消费者线程从队列中读取数字并进行加和运算，统计出所有数字的和。

生产者线程和消费者线程使用了线程同步技术，通过使用互斥锁的方式，确保了对全局队列的同步访问。

同时，通过信号量semaphore，消费者线程在等待队列中有数字时可以被唤醒，从而实现了线程间通信。

## 心得体会

这次作业让我对于进程和线程有了更深的理解。线程的出现是为了提高程序执行的效率，它的量级要小于进程。一个进程的多个线程共享同一个虚拟地址空间。多个线程对于全局数据的访问需要采用锁的方式，以防止数据的不同步。信号量可以实现进程之间的通信。总的来说，我受益匪浅。