Университет ИТМО Кафедра ИПМ

Отчет по лабораторной работе № 2

«Синхронизация потоков при помощи семафоров и критических секций»

Выполнил: студент группы P3317

Плюхин Д.А.

Преподаватель: Лаздин Артур Вячеславович

Санкт-Петербург
2017 год

Задание

Написать программу для консольного процесса, который состоит из трёх потоков:
Поток **main** должен выполнить следующие действия:

создать массив, размерность и элементы которого вводятся пользователем с консоли;
вывести размерность и элементы исходного массива на консоль;
ввести число k;
запустить поток **work**;
запустить поток **SumElement**;
освобождение выходной поток stdout после вывода на консоль каждого нового элемента массива.

выводить на экран поэлементно элементы массива (итогового) параллельно с работой потока **work**;
известить поток **SumElement** о начале суммирования (момент запуска произойдёт после того, будут готовы к печати k - элементов массива).

Поток **work** должен выполнить следующие действия *(Для синхронизации с потоком **main** – использовать **семафор. Проверить работу программы используя критическую секцию для** синхронизации с потоком **main, объяснить отличия, если есть!**)*:

запросить у пользователя временной интервал, требуемый для отдыха после подготовки одного элемента в массиве;
Поиск в массиве простых чисел (разместить их в массиве слева, остальные элементы массива - справа).
Элементы - целые числа без знака.
извещать поток main о новом элементе;
после каждого готового элемента отдыхать в течение заданного интервала времени;

Поток **SumElement** должен выполнить следующие действия (*Для синхронизации с потоком main, использовать бинарный семафор!*):

ждёт от потока **main** сигнал о начале суммирования;
выполнить суммирование элементов итогового массива до заданной позиции k; вывести итоговую сумму.

Исходный код

```
using System;

using System.Threading;

namespace FirstLab
{
    class Threads
    {

        static void Main()
        {
            Thread mainThread = new Thread(main);
            mainThread.Start();
        }

        static void main()
        {
            Semaphore main_work_semaphore;
            Semaphore main_sum_element_semaphore;

            main_work_semaphore = new Semaphore(0, 1);
            main_sum_element_semaphore = new Semaphore(0, 1);

            int arraySize = getIntFromUser("Please, type size of array:");
            uint[] array = new uint[arraySize];

            for (int i = 0; i < arraySize; i++){
              array[i] = getUintFromUser("Please, type "+(i+1)+" element of array:");
            }

            showArray(array);

            int k = getIntFromUser("Please, type k:");

            //Create threads
            Thread workThread = new Thread(() => work(ref array, k, ref main_work_semaphore));
```

```
            Thread sumElementThread = new Thread(() => SumElement(ref array, k, ref
main_sum_element_semaphore));

            //Run threads
            workThread.Start();
            sumElementThread.Start();

            int current_index = 0;
            while (current_index < k){
              main_work_semaphore.WaitOne();
              Console.WriteLine("main: "+(current_index+1)+" element is known: "+array[current_index]);
              current_index++;
            }
            Console.WriteLine("main: "+k+" elements are known, so signaling to the SumElement...");
            main_sum_element_semaphore.Release();
        }

        static void work(ref uint[] array, int k, ref Semaphore main_work_semaphore)
        {
            int rest_interval = getIntFromUser("Please, type rest interval (milliseconds):");
            for (int i = 0; i < array.Length; i++){
              int selected_element = i;
              for (int j = i; j < array.Length; j++){
                if (isSimple(array[j])){
                  selected_element = j;
                  break;
                }
              }
              uint tmp = array[i];
              array[i] = array[selected_element];
              array[selected_element] = tmp;
              Console.WriteLine("work: "+(i+1)+" element is ready!");
              if (i < k) main_work_semaphore.Release();
              Thread.Sleep(rest_interval);
            }
        }

        static void SumElement(ref uint[] array, int k, ref Semaphore main_sum_element_semaphore)
        {
            main_sum_element_semaphore.WaitOne();
            Console.WriteLine("SumElement: Sum of first "+k+" elements of array is "+countSum(array,k));
        }

        static void showArray(uint[] array){
          Console.Write("Given array with size "+array.Length+" and elements ");
          for (int i = 0; i < array.Length; i++){
            if (i != 0) Console.Write(", ");
            Console.Write(array[i]);
          }
          Console.WriteLine();
        }

        static uint countSum(uint[] array, int k){
          uint sum = 0;
          for (int i = 0; i < k; i++){
            sum += array[i];
          }
          return sum;
        }

        static bool isSimple(uint number){
          if (number < 2) return true;
          for (uint i = 2; i < (uint)(Math.Floor(Math.Sqrt(number)) + 1); i++){
            if (0 == number % i) return false;
          }
          return true;
        }

        static uint getUintFromUser(string msg){
          Console.WriteLine(msg);
          return Convert.ToUInt32(Console.ReadLine());
        }

        static int getIntFromUser(string msg){
```

```
            Console.WriteLine(msg);
            return Convert.ToInt32(Console.ReadLine());
        }
    }
}
```

Результат

```
Please, type size of array:
7
Please, type 1 element of array:
23
Please, type 2 element of array:
12
Please, type 3 element of array:
89
Please, type 4 element of array:
93
Please, type 5 element of array:
1
Please, type 6 element of array:
1
Please, type 7 element of array:
1
Given array with size 7 and elements 23, 12, 89, 93, 1, 1, 1
Please, type k:
6
Please, type rest interval (milliseconds):
1000
work: 1 element is ready!
main: 1 element is known: 23
work: 2 element is ready!
main: 2 element is known: 89
work: 3 element is ready!
main: 3 element is known: 1
work: 4 element is ready!
main: 4 element is known: 1
work: 5 element is ready!
main: 5 element is known: 1
work: 6 element is ready!
main: 6 element is known: 93
main: 6 elements are known, so signaling to the SumElement...
SumElement: Sum of first 6 elements of array is 208
work: 7 element is ready!
```

Вывод

Таким образом, взаимодействие потоков достаточно просто реализуется при помощи таких объектов синхронизации, как семафоры, и, в частности, бинарные семафоры, которые, по сути, ничем не отличаются от критических секций, и именно по этой причине в данном случае реализация взаимодействия потоков work и main не принципиальна и может быть выполнена равноценно как с помощью семафора, так и с помощью критической секции, поскольку оба варианта синхронизации предусматривают использование двоичной блокирующей переменной, операции изменения которой атомарны.