Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра ИПМ

**«Операционные системы»**

**Лабораторная работа № 3**

Вариант № 11

Выполнил: Сорокин Ю.Б.

Группа: Р3317

Преподаватель: Лаздин А.В.

Санкт-Петербург

2016 год

**Цель работы**

Написать программу для консольного процесса, который состоит из трёх потоков: main , work, и третьего.

**Задание**

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 выведены на консоль k элементов массива).

Поток work должен выполнить следующие действия (Для синхронизации с потоком main – использовать

Семафор):

запросить у пользователя временной интервал, требуемый для отдыха после подготовки одного элемента в массиве;

Приведение массива к палиндрому (получившейся палиндром поместить в массиве слева, а лишние элементы соответственно – справа ) Элементы - символы

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 критическую секцию!):

ждёт от потока main сигнал о начале суммирования;

выполнить суммирование элементов (кодов) итогового массива до заданной позиции k;

вывести итоговую сумму.

**Код программы**

**public class** main {  
 **public static void** main(String args[]) **throws** InterruptedException {  
 ArrayList<Integer> mas = **new** ArrayList<>();  
 Scanner in = **new** Scanner(System.***in***);  
 Semaphore MainSem = **new** Semaphore(0);  
 Semaphore WorkSem = **new** Semaphore(0);  
 Object obj = **new** Object();  
  
 System.***out***.println(**"Размерность массива"**);  
 **int** n = in.nextInt();  
 System.***out***.println(**"Число K"**);  
 **int** k = in.nextInt();  
 **for**(**int** i=0;i<n;i++){  
 mas.add((4\*i+11)/5);  
 System.***out***.print(mas.get(i)+**" "**);  
 }  
 System.***out***.println();  
  
 Thread work = **new** Thread(**new** work(MainSem,WorkSem,mas,**"Work thread"**));  
 Thread sum = **new** Thread(**new** Sum(mas,obj,k,**"Sum thread"**));  
  
 work.start();  
  
 **int** index=0;  
 **while**(index!=(mas.size())) {  
 MainSem.acquire();  
 **synchronized** (obj) {  
 System.***out***.print(mas.get(index) + **" "**);  
 }  
 MainSem.release();  
 WorkSem.acquire();  
 index++;  
 **if**((index+1)==k) sum.start();  
 }  
 }

**public class** work **implements** Runnable {  
 Semaphore **MainSem**;  
 Semaphore **WorkSem**;  
 ArrayList<Integer> **list**;  
 String **name**;  
  
 work(Semaphore MainSem,Semaphore WorkSem,ArrayList<Integer> list, String name){  
 **this**.**list**=list;  
 **this**.**MainSem**=MainSem;  
 **this**.**WorkSem**=WorkSem;  
 **this**.**name**= name;  
 }  
  
 **public void** run(){  
 Scanner in = **new** Scanner(System.***in***);  
 **int** time=0;  
 **WorkSem**.release();  
 **try** {  
 **WorkSem**.acquire();  
 System.***out***.println(**"Время ожидания"**);  
 time = in.nextInt();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
  
 **int** botMarker=0;  
 **int** watched = 0;  
 **int** topMarker = 0;  
 **while**(watched!=**list**.size()){  
 **int** cur = **list**.get(topMarker);  
 **int** indexOf = **list**.lastIndexOf(cur);  
 **if**(topMarker!=indexOf){  
 **list**.remove(topMarker);  
 **list**.remove(indexOf-1);  
 **list**.add(botMarker,cur);  
  
 notifyMain(time);  
  
 **list**.add(botMarker,cur);  
 botMarker++;  
 topMarker+=2;  
 watched++;  
 }  
 **else**{  
 **list**.add(cur);  
 **list**.remove(topMarker);  
 }  
 watched++;  
 }  
  
 **while**(botMarker!=**list**.size()+1){  
 notifyMain(time);  
 botMarker++;  
 }  
  
 }  
 **void** notifyMain(**int** time){  
 **MainSem**.release();  
 **try** {  
 Thread.*sleep*(time\*1000);  
 **MainSem**.acquire();  
 **WorkSem**.release();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

**public class** Sum **implements** Runnable {  
 String **name**;  
 ArrayList<Integer> **mas**;  
 **int sum**=0;  
 Object **obj**;  
 **int k**;  
  
 **public** Sum(ArrayList<Integer> mas, Object obj,Integer k, String name){  
 **this**.**k**=k;  
 **this**.**obj**=obj;  
 **this**.**mas**=mas;  
 **this**.**name** = name;  
 }  
 **public void** run(){  
 **synchronized** (**obj**) {  
 **for**(**int** i=0;i<**k**;i++){  
 **sum** += **mas**.get(i);  
 }  
 System.***out***.println();  
 System.***out***.println(**"Сумма равна "** + **sum**);  
 }  
 }  
}

**Пример результата запуска программы**

Размерность массива

10

Число K

3

2 3 3 4 5 6 7 7 8 9

Время ожидания

1

3 7 7

Сумма равна 17

3 2 4 5 6 8 9

**Вывод**

При выполнении данной лабораторной работы я ознакомился с различными способами синхронизации потоков