МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

**Домашнее задание**

|  |  |  |
| --- | --- | --- |
| по дисциплине | Операционные системы |  |

|  |  |
| --- | --- |
| Студента | Абрамова Егора Владимировича |
|  | фамилия, имя, отчество полностью |
| Курс | 1 Группа ФИТ-221 |
| Направление | 02.03.02 Фундаментальная информатика |
|  | и информационные технологии |
|  | код, наименование |
| Руководитель | Старший преподаватель |
|  | должность, ученая степень, звание |
|  | Карабцов Р.Д |
|  | фамилия, инициалы |
| Выполнил | 7.05.2024 |
|  | дата, подпись студента |
| баллы |  |
|  | дата, подпись руководителя |

Омск-2024

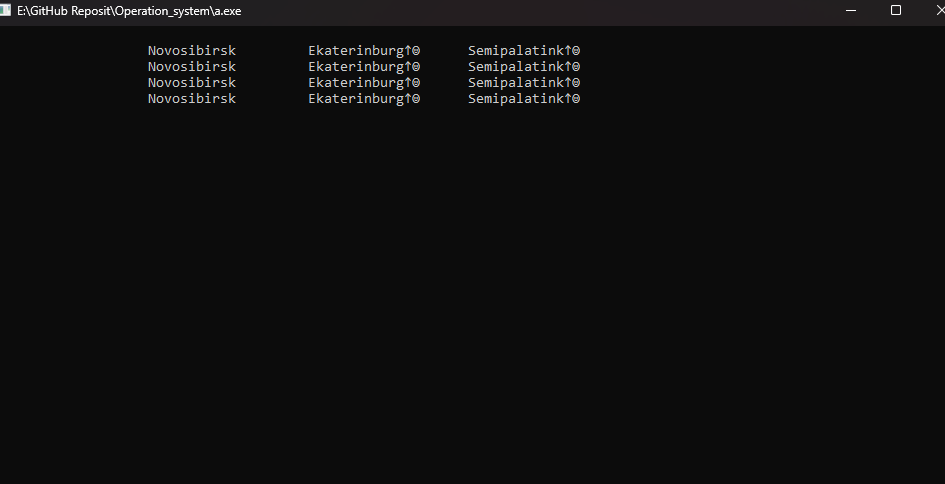
**Задание**

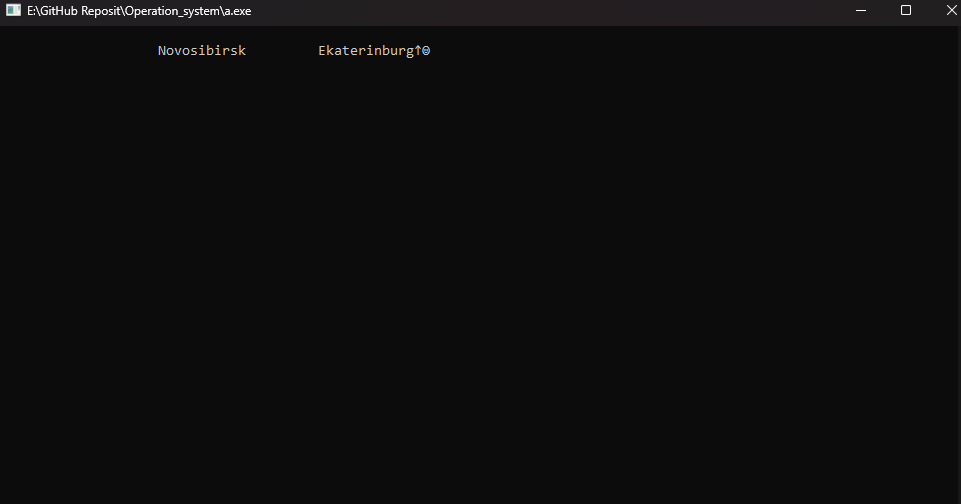
Задание Разработать многопоточную программу, отображающие на экране взаимодействие трех нитей «читателей» из общей области данных и трех «писателей», записывающих в этот буфер данные. Буфер предназначен для хранения 12 символов. Первая нить-писатель выводит латинскими буквами название города "Novosibirsk", вторая нить-писатель выводит латинскими буквами название города "Semipalatink", а третья — название города "Ekaretinburg". Такой вывод эти три нити осуществляют в два приема, первый из которых записывает половину своего текста без завершающего этот промежуточный текст нуля. Между такими половинами вывода нити производят задержку на случайную величину миллисекунд, но не более 1 сек. После вывода своего текста в буфер каждая нить-писатель переходит в ожидание порядка 2-3 сек до следующей попытки записи в буфер. Нити-читатели - через случайный интервал порядка 300 мсек - читают данные из буфера, если это позволяют средства синхронизации доступа между нитями, и вывод прочитанный текст на экран, каждая в свой столбец. Каждый вывод нити-читателя осуществляется в новую строку своего столбца, поэтому суммарные действия вывода в таких нитях предусмотреть только для 20 - 24 строк. Синхронизацию осуществить с помощью семафоров.

1. Чем абстрактные семафоры отличаются от известных вам программных реализаций.

2. Укажите в каких ситуациях предпочтительней использование мьютексов вместо семафоров и в каких ситуациях целесообразно обратное предпочтение

**Решение**

*Рис 1. Вывод программы*



*Рис 2. Вывод программы*

**Ответы**

1.Отличие абстрактных семафоров от программных реализаций: Программные реализации: Обычно это конкретные механизмы синхронизации, такие как мьютексы, семафоры, условные переменные и прочее, предоставляемые операционной системой или библиотекой. Эти реализации обычно имеют явный API и связаны с конкретной платформой. Абстрактные семафоры: Это более высокоуровневые абстракции, предоставляющие удобный интерфейс для управления доступом к ресурсам или выполнением потоков. Они не обязательно привязаны к конкретным механизмам операционной системы и предоставляют общие концепции для управления параллельным выполнением.

2.Выбор между мьютексами и семафорами: Мьютексы: Предпочтительны, когда нужно обеспечить эксклюзивный доступ к ресурсу: Если важно, чтобы только один поток одновременно мог использовать определенный ресурс или выполнить критическую секцию кода, мьютексы являются хорошим выбором. Простота использования: Мьютексы обычно проще в использовании, чем семафоры, и легче поддерживают основные сценарии блокировки. Обеспечение взаимного исключения: Мьютексы часто используются для создания критических секций, где только один поток может выполнять код в определенный момент времени. Семафоры: Управление ресурсами с ограниченной емкостью: Если нужно управлять доступом к ресурсам, которые могут использовать несколько потоков одновременно (например, пул потоков), семафоры с ограниченной емкостью могут быть более подходящими. Синхронизация между несколькими потоками: Семафоры могут использоваться для синхронизации между потоками, не обязательно ограничивая количество одновременно активных потоков. Сложные сценарии блокировки: В более сложных сценариях, где нужно управлять группами потоков, а не просто предоставлять или ограничивать доступ к ресурсам, семафоры могут быть более гибкими

**Листинг программы**

#include <windows.h>  
#include <process.h>  
#include <stdio.h>  
#include <conio.h>  
#include <time.h>  
  
HANDLE hthread[6], hsm, hsm1;  
char buffer[12], buff[12];  
  
void GoToXY(const int X, const int Y) {  
 HANDLE OutputHandle;  
 CONSOLE\_SCREEN\_BUFFER\_INFO ScreenBufInfo;  
 OutputHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);  
 GetConsoleScreenBufferInfo(OutputHandle, &ScreenBufInfo);  
 ScreenBufInfo.dwCursorPosition.X = X;  
 ScreenBufInfo.dwCursorPosition.Y = Y;  
 SetConsoleCursorPosition(OutputHandle, ScreenBufInfo.dwCursorPosition);  
}  
  
DWORD WINAPI Writer(void\* n) {  
 short i;  
 while (1) {  
 WaitForSingleObject(hsm, 3000);  
 if ((int)n == 3) {  
 strncpy(buff, "Novosibirsk", 12);  
 } else if ((int)n == 4) {  
 strncpy(buff, "Ekaterinburg", 12);  
 } else if ((int)n == 5) {  
 strncpy(buff, "Semipalatink", 12);  
 }  
 for (i = 0; i <= 5; i++) {  
 buffer[i] = buff[i];  
 }  
 Sleep(1000);  
 for (i = 6; i <= 11; i++) {  
 buffer[i] = buff[i];  
 }  
 ReleaseSemaphore(hsm1, 1, NULL);  
 }  
 return 0;  
}  
  
DWORD WINAPI Reader(void\* n) {  
 short i = 0, j;  
 while (1) {  
 i++;  
 WaitForSingleObject(hsm1, INFINITE);  
 GoToXY(((int)n + 1) \* 20, i);  
 printf("%s", buffer);  
 ReleaseSemaphore(hsm, 1, NULL);  
 Sleep(290);  
 }  
 return 0;  
}  
  
void main() {  
 int i;  
 system("cls");  
 hsm = CreateSemaphore(NULL, 0, 1, NULL);  
 hsm1 = CreateSemaphore(NULL, 0, 1, NULL);  
 for (i = 3; i <= 5; i++) {  
 hthread[i] = CreateThread(NULL, 4096, Writer, (void\*)i, 0, NULL);  
 }  
 for (i = 0; i <= 2; i++) {  
 hthread[i] = CreateThread(NULL, 4096, Reader, (void\*)i, 0, NULL);  
 }  
 ReleaseSemaphore(hsm, 1, NULL);  
 getchar();  
 for (i = 0; i <= 5; i++) CloseHandle(hthread[i]);  
 CloseHandle(hsm);  
}

**Ниже приведена блок-схема алгоритма.**

