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 | 8.04.2024 |
|  | дата, подпись студента |
| баллы |  |
|  | дата, подпись руководителя |

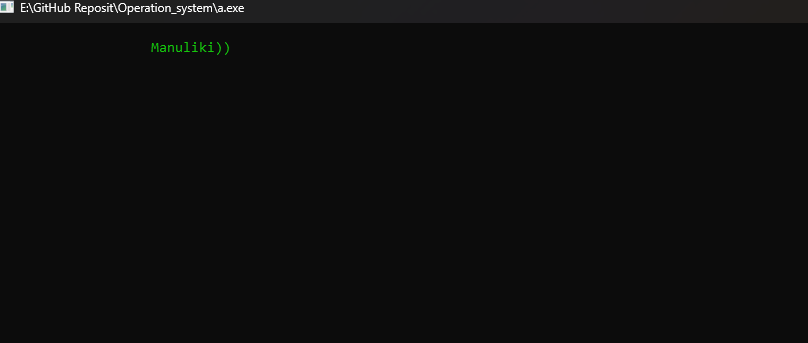
Омск-2024

**Задание.** Разработать программу с тремя дополнительными нитями (threads) относительно главной нити. Каждая из нитей должна использовать общие для всех нитей данные, представленные массивом символов, в которых записаны 20 первых букв латинского алфавита. Каждая из этих нитей на своем k-м шаге выводит со своей случайной задержкой на место «своего» столбца экрана k-ю букву из указанного массива латинских букв, причем с числом повторений, равному условному номеру нити, умноженному на два. Каждая из используемых нитей должен осуществлять вывод своим цветом, отличным от остальных нитей. На 6-м шаге главная нить делает попытку отмены первой из дополнительных нитей, а на 11-м делает попытку отмены третьей из дополнительных нитей. Первая и третья дополнительная нити в начале своей работы запрещают свою отмену. Третья нить на 13 шаге разрешает отмену, но в отложенном режиме. Точку отмены эта нить устанавливает между 16 и 17-м шагом своей работы. Все управляющие указания должны отображаться сообщениями без прокрутки экрана (в фиксированные позиции экрана). (Базовый вариант.) Первая и третья дополнительная нити должны использовать общую процедуру, которая исходя из информации своего аргумента, поступившей в нее при запуске, принимает правильные решения о горизонтальной позиции своего вывода и цвете вывода. (Дополнительное задание для максимального рейтинга).

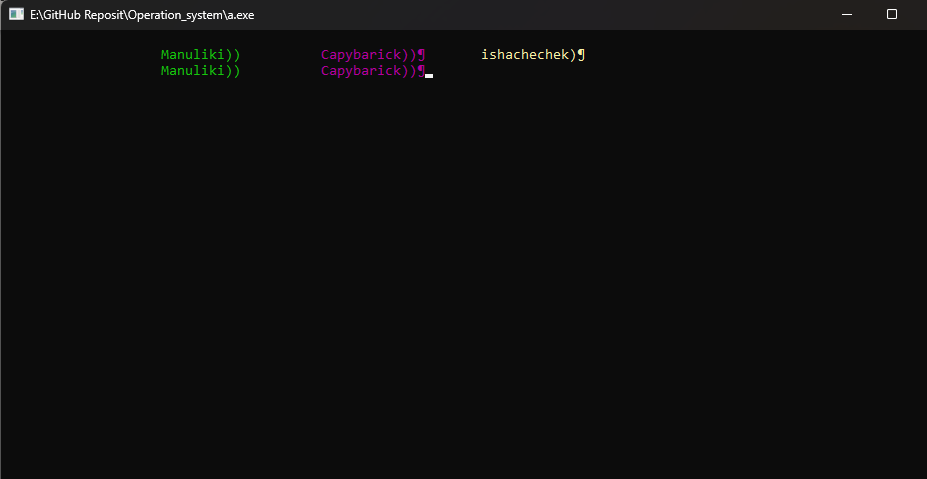
**Листинг программы**

#include <stdio.h>  
#include <stdlib.h>  
#include <windows.h>  
#include <conio.h>  
  
HANDLE consoleout;  
HANDLE Semaphore, Semaphore2;  
CRITICAL\_SECTION CriticalSection;  
int Delay1, Delay2;  
char Buffer[12];  
char names[3][16] = {  
 {"Manuliki))"},  
 {"Capybarick))"},  
 {"ishachechek))"}};  
int n = 0;  
void SetWindow(int Width, int Height)  
{  
 COORD coord;  
 coord.X = Width;  
 coord.Y = Height;  
  
 SMALL\_RECT Rect;  
 Rect.Top = 0;  
 Rect.Left = 0;  
 Rect.Bottom = Height - 1;  
 Rect.Right = Width - 1;  
  
 SetConsoleScreenBufferSize(consoleout, coord);  
 SetConsoleWindowInfo(consoleout, TRUE, &Rect);  
}  
DWORD WINAPI Write(void \*arg)  
{  
 int r = (int)arg;  
 r--;  
  
 while (1)  
 {  
 WaitForSingleObject(Semaphore, INFINITE);  
  
 for (n = 0; n < 6; n++)  
 {  
 Buffer[n] = names[r][n];  
 }  
 Sleep(100 \* (int)Delay1);  
  
 for (n = 6; n < 12; n++)  
 {  
 Buffer[n] = names[r][n];  
 }  
 ReleaseSemaphore(Semaphore2, 1, NULL);  
 Sleep(2000 + rand() % 1001);  
 }  
}  
DWORD WINAPI Read(void \*arg)  
{  
 COORD pos;  
 char buff[12];  
 int lenBuff;  
 for (int i = 0; i < 40; i++)  
 {  
 WaitForSingleObject(Semaphore2, INFINITE);  
 pos.X = 20 \* (int)arg;  
 pos.Y = i + 1;  
 EnterCriticalSection(&CriticalSection);  
 SetConsoleCursorPosition(consoleout, pos);  
  
 switch((int)arg){  
 case 1: SetConsoleTextAttribute(consoleout, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);  
 break;  
 case 2: SetConsoleTextAttribute(consoleout, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);  
 break;  
 case 3: SetConsoleTextAttribute(consoleout, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);  
 break;  
 }  
 lenBuff = sprintf(buff, "%s", Buffer);  
 lenBuff = ((int)arg > 1) ? lenBuff - 1 : lenBuff;  
 WriteFile(consoleout, buff, lenBuff, NULL, NULL);  
 LeaveCriticalSection(&CriticalSection);  
 ReleaseSemaphore(Semaphore, 1, NULL);  
 Sleep(100 \* (int)Delay2);  
 }  
 return 0;  
}  
  
void main(int args, char \*argv[])  
{  
 system("cls");  
 consoleout = GetStdHandle(STD\_ERROR\_HANDLE);  
 SetWindow(120, 30);  
 HANDLE writeThread[3];  
 HANDLE readThread[3];  
 Delay1 = 20;  
 Delay2 = 3;  
 InitializeCriticalSection(&CriticalSection);  
  
 if (args >= 3)  
 {  
 sscanf(argv[1], "%d", &Delay1);  
 sscanf(argv[2], "%d", &Delay2);  
 }  
 Semaphore = CreateSemaphore(NULL, 1, 1, NULL);  
 Semaphore2 = CreateSemaphore(NULL, 0, 1, NULL);  
  
 for (int i = 0; i < 3; i++)  
 {  
 writeThread[i] = CreateThread(NULL, 4096, Write, (void \*)(i + 1), 0, NULL);  
 readThread[i] = CreateThread(NULL, 4096, Read, (void \*)(i + 1), 0, NULL);  
 }  
 getchar();  
 DeleteCriticalSection(&CriticalSection);  
  
 for (int i = 0; i < 3; i++)  
 {  
 CloseHandle(writeThread[i]);  
 CloseHandle(readThread[i]);  
 }  
}

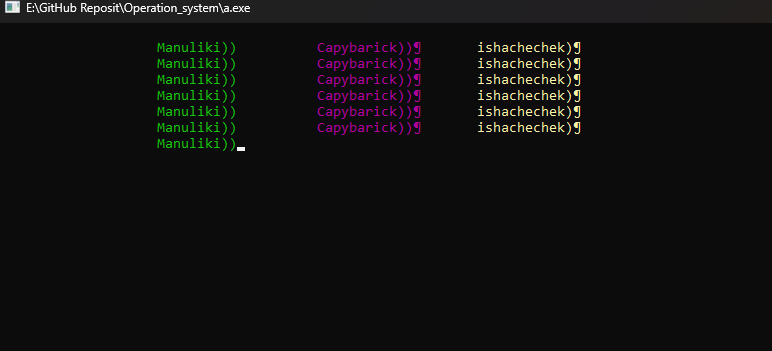
**Примеры работы программы**

****

*Рис 1. Первый скрин*

****

*Рис 2. Второй скрин*

****

*Рис 3. Третий скрин*