Лабораторная работа № 3

по курсу Операционные системы:

Системные вызовы и работа с потоками

Выполнил студент группы М80-204Б МАИ Дюсекеев Алишер

Оценка_____

Лабораторная работа №3

Цель работы

Целью является приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). При создании необходимо предусмотреть ключи, которые позволяли бы задать максимальное количество потоков, используемое программой. При возможности необходимо использовать максимальное количество возможных потоков. Ограничение потоков может быть задано или ключом запуска вашей программы, или алгоритмом.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

Вариант задания № 16

Наложить K раз фильтры эрозии и наращивания на матрицу состоящую из вещественных чисел. На выходе получается 2 результирующие матрицы.
Вариант лабораторной работы выдается преподавателем.

Содержание main.cpp

```
// ConsoleApplication2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <windows.h>
#include <stdlib.h>
#include <math.h>

#define MAX_THREADS 1
#define K 5


double ardoub[K][K] = { { 1,1,0,1,1 },{ 1, 3, 2.6, 1, 1 },{ 1,1,1,1,1 },{ 1,1,1,1,1 },{ 1,1,1,1,1 } },
ardouber[K][K], ardoubc[K][K], ardoubpri[K][K];
HANDLE mutex, mutex2;

DWORD WINAPI ThreadFunction(LPVOID lpParam)
{
	int myNumber = (int)lpParam;

	WaitForSingleObject(mutex, INFINITE);
	for (int i = 0; i < K; i++) {
		for (int j = 0; j < K; j++) {
			if (ardoub[i][j] > 1) {
				if (i != 0 && i != K - 1 && j != 0 && j != K - 1 && (ardoub[i + 1][j] == 0
|| ardoub[i - 1][j] == 0 || ardoub[i][j - 1] == 0 || ardoub[i][j + 1] == 0)) {
					ardouber[i][j] = ardoub[i][j] - 1;
				}
				else if (i == 0 || i == K - 1 || j == 0 || j == K - 1) {
					ardouber[i][j] = ardoub[i][j] - 1;
				}
				else {
					ardouber[i][j] = ardoub[i][j];
				}
			}
			else if (ardoub[i][j] <= 1) {
				if (i != 0 && i != K - 1 && j != 0 && j != K - 1 && (ardoub[i + 1][j] == 0
|| ardoub[i - 1][j] == 0 || ardoub[i][j - 1] == 0 || ardoub[i][j + 1] == 0)) {
					ardouber[i][j] = 0;
				}
				else if (i == 0 || i == K - 1 || j == 0 || j == K - 1) {
					ardouber[i][j] = 0;
```

```
                                }
                                else {
                                        ardouber[i][j] = ardoub[i][j];
                                }
                        }
                }
        }


        for (int i = 0; i < K; i++) {
                for (int j = 0; j < K; j++) {
                        ardoub[i][j] = ardouber[i][j];
                }
        }


        ReleaseMutex(mutex);
        return myNumber;
}

DWORD WINAPI ThreadFunction2(LPVOID lpParam)
{
        int myNumber = (int)lpParam;

        WaitForSingleObject(mutex2, INFINITE);
        for (int i = 0; i < K; i++) {
                for (int j = 0; j < K; j++) {
                        if (ardoubc[i][j] != 0) {
                                if (i != 0 && i != K - 1 && j != 0 && j != K - 1) {
                                        //(ardoub[i + 1][j] == 0 || ardoub[i - 1][j] == 0 || ardoub[i][j -
1] == 0 || ardoub[i][j + 1] == 0)
                                        ardoubpri[i + 1][j] = ardoubc[i + 1][j] + 1;
                                        ardoubpri[i - 1][j] = ardoubc[i - 1][j] + 1;
                                        ardoubpri[i][j + 1] = ardoubc[i][j + 1] + 1;
                                        ardoubpri[i][j - 1] = ardoubc[i][j - 1] + 1;
                                }
                                else if (i == 0 || i == K - 1 || j == 0 || j == K - 1) {
                                        if (i == 0 && j == 0) {
                                                ardoubpri[i + 1][j] = ardoubc[i + 1][j] + 1;
                                                ardoubpri[i][j + 1] = ardoubc[i][j + 1] + 1;
                                        }
                                        else
                                                if (i == 0 && j == K - 1) {
                                                        ardoubpri[i + 1][j] = ardoubc[i + 1][j] + 1;
                                                        ardoubpri[i][j - 1] = ardoubc[i][j - 1] + 1;
                                                }
                                                else
                                                        if (i == K - 1 && j == 0) {
                                                                ardoubpri[i - 1][j] = ardoubc[i - 1][j] +
1;

                                                                ardoubpri[i][j + 1] = ardoubc[i][j + 1] +
1;

                                                        }
                                                        else
                                                                if (i == K - 1 && j == K - 1) {
                                                                        ardoubpri[i - 1][j] = ardoubc[i -
1][j] + 1;

                                                                        ardoubpri[i][j - 1] =
ardoubc[i][j - 1] + 1;

                                                                }
                                                                else
                                                                        if (i == 0) {
                                                                                ardoubpri[i + 1][j] =
ardoubc[i + 1][j] + 1;

                                                                                ardoubpri[i][j + 1] =
ardoubc[i][j + 1] + 1;

                                                                                ardoubpri[i][j - 1] =
ardoubc[i][j - 1] + 1;

                                                                        }
                                                                        else
                                                                                if (i == K - 1) {
                                                                                        ardoubpri[i -
1][j] = ardoubc[i - 1][j] + 1;

                                                                                        ardoubpri[i][j +
1] = ardoubc[i][j + 1] + 1;

                                                                                        ardoubpri[i][j -
1] = ardoubc[i][j - 1] + 1;
```

```
                                                                                }
                                                                                else
                                                                                        if (j == 0) {

                ardoubpri[i + 1][j] = ardoubc[i + 1][j] + 1;

                ardoubpri[i - 1][j] = ardoubc[i - 1][j] + 1;

                ardoubpri[i][j + 1] = ardoubc[i][j + 1] + 1;
                                                                                        }
                                                                                        else
                                                                                                if (j ==
K - 1) {

                ardoubpri[i + 1][j] = ardoubc[i + 1][j] + 1;

                ardoubpri[i - 1][j] = ardoubc[i - 1][j] + 1;

                ardoubpri[i][j - 1] = ardoubc[i][j - 1] + 1;
                                                                                                }
                                        }
                                        else {
                                                ardoubpri[i][j] = ardoubc[i][j];
                                        }
                                }
                        }
                }


                for (int i = 0; i < K; i++) {
                        for (int j = 0; j < K; j++) {
                                ardoubc[i][j] = ardoubpri[i][j];
                        }
                }


                ReleaseMutex(mutex2);
                return myNumber;
}



int main()
{
        HANDLE  hThreadArray[MAX_THREADS];
        DWORD   dwThreadIdArray[MAX_THREADS];
        mutex = CreateMutex(NULL, FALSE, NULL);
        mutex2 = CreateMutex(NULL, FALSE, NULL);

        printf("Matrix\n");

        for (int i = 0; i < K; i++) {
                for (int j = 0; j < K; j++) {
                        printf("%.3f ", ardoub[i][j]);
                }
                printf("\n");
        }
        printf("\n");

        for (int i = 0; i < K; i++) {
                for (int j = 0; j < K; j++) {
                        ardoubc[i][j] = ardoub[i][j];
                }
        }

        for (int i = 0; i < MAX_THREADS; i++)
        {
                hThreadArray[i] = CreateThread(NULL,
                        0,
                        ThreadFunction,
                        (LPVOID)i,
                        CREATE_SUSPENDED,
                        &dwThreadIdArray[i]);
        }
        for (int i = 0; i < MAX_THREADS; i++) ResumeThread(hThreadArray[i]);

        WaitForMultipleObjects(MAX_THREADS, hThreadArray, TRUE, INFINITE);
```

```
	for (int i = 0; i < MAX_THREADS; i++) CloseHandle(hThreadArray[i]);

	for (int i = 0; i < MAX_THREADS; i++)
	{
		hThreadArray[i] = CreateThread(NULL,
			0,
			ThreadFunction2,
			(LPVOID)i,
			CREATE_SUSPENDED,
			&dwThreadIdArray[i]);
	}

	for (int i = 0; i < MAX_THREADS; i++) ResumeThread(hThreadArray[i]);

	WaitForMultipleObjects(MAX_THREADS, hThreadArray, TRUE, INFINITE);
	for (int i = 0; i < MAX_THREADS; i++) CloseHandle(hThreadArray[i]);

	printf("Erosia\n");

	for (int i = 0; i < K; i++) {
		for (int j = 0; j < K; j++) {
			printf("%.3f ", ardouber[i][j]);
		}
		printf("\n");
	}
	printf("\n");
	printf("Prirashenie\n");
	for (int i = 0; i < K; i++) {
		for (int j = 0; j < K; j++) {
			printf("%.3f ", ardoubpri[i][j]);
		}
		printf("\n");
	}

	system("pause");
	return 0;
}
```

Использованные системные вызовы.

Работа с мютексом и потоками.

-Создание мютекса.

```
HANDLE WINAPI CreateMutex(
  _In_opt_ LPSECURITY_ATTRIBUTES lpMutexAttributes,
  _In_     BOOL                  bInitialOwner,
  _In_opt_ LPCTSTR               lpName
);
```

-Создание потока

```
HANDLE WINAPI CreateThread(
  _In_opt_  LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  _In_      SIZE_T                 dwStackSize,
  _In_      LPTHREAD_START_ROUTINE lpStartAddress,
  _In_opt_  LPVOID                 lpParameter,
  _In_      DWORD                  dwCreationFlags,
  _Out_opt_ LPDWORD                lpThreadId
);
```

-Приостановка потока.

```
DWORD WINAPI WaitForSingleObject(
  _In_ HANDLE hHandle,
  _In_ DWORD  dwMilliseconds
);
```

-Возобновление потока

```
BOOL WINAPI ReleaseMutex(
  _In_ HANDLE hMutex
);
```

Вывод

Обработка программ в много поточном режиме позволяет быстро и эффективно распараллеливать нагрузку на несколько процессоров. Их создание и контроль производится благодаря функциям из win32api и linuxapi. Для взаимодействия потоков можно использовать такие вещи как mutex, semaphore, барьер. Это нужно для того чтобы потоки не лезли в переменные пока с ними работают другие потоки. Можно реализовать это и с помощью специальных алгоритмов. Также можно создавать определенное количество потоков и ограничивать их с помощью контрольных значений.