Бруцкая Анастасия

Лабораторная работа №1

TaskQueue.cs

using System.Collections.Concurrent;

namespace lapa1;

public delegate void TaskDelegate();

public class TaskQueue

{

private Thread[] threads;

private BlockingCollection<TaskDelegate> taskQueue;

private bool running;

public TaskQueue(int threadCount)

{

threads = new Thread[threadCount];

taskQueue = new BlockingCollection<TaskDelegate>();

running = true;

for (int i = 0; i < threadCount; i++)

{

threads[i] = new Thread(RunTasks)

{

//object initializer

IsBackground = true

};

threads[i].Start();

}

}

private void RunTasks()

{

foreach (var task in taskQueue.GetConsumingEnumerable())

{

try

{

task();

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка выполнения задачи: {ex.Message}");

}

}

}

public void AddTask(TaskDelegate task)

{

if (!running)

{

throw new InvalidOperationException("Thread pool is stopped.");

}

taskQueue.Add(task);

}

public void Stop()

{

running = false;

taskQueue.CompleteAdding();

foreach (var thread in threads)

{

thread.Join();

}

}

}

Task1.cs

namespace lapa1;

class Program

{

public static void Main(string[] args)

{

string? s = Console.ReadLine();

int threadCount = s == null ? 0 : int.Parse(s);

TaskQueue taskQueue = new TaskQueue(threadCount);

for (int i = 0; i < 10; i++)

{

int taskNumber = i;

taskQueue.AddTask(() =>

{

Console.WriteLine($"Running task {taskNumber} on thread {Thread.CurrentThread.ManagedThreadId}");

});

}

Thread.Sleep(2000);

taskQueue.Stop();

}

}

Task1.2.cs

using System.Diagnostics;

namespace lapa1.\_2;

class Task2

{

public static void Main(string[] args)

{

if (args.Length < 2)

{

Console.WriteLine("Использование: copyapp <исходный каталог> <целевой каталог>");

return;

}

string sourceDir = args[0];

string targetDir = args[1];

if (!Directory.Exists(sourceDir))

{

Console.WriteLine($"Ошибка: Исходный каталог '{sourceDir}' не существует.");

return;

}

Directory.CreateDirectory(targetDir);

Console.WriteLine($"[INFO] Копирование файлов из '{sourceDir}' в '{targetDir}'...");

// Запуск таймера

Stopwatch stopwatch = Stopwatch.StartNew();

CopyFilesParallel(sourceDir, targetDir);

stopwatch.Stop();

Console.WriteLine($"[INFO] Время выполнения: {stopwatch.Elapsed.TotalSeconds:F2} секунд");

}

static void CopyFilesParallel(string sourceDir, string targetDir)

{

string[] files = Directory.GetFiles(sourceDir, "\*", SearchOption.AllDirectories);

int copiedCount = 0;

Parallel.ForEach(files, new ParallelOptions { MaxDegreeOfParallelism = Environment.ProcessorCount }, file =>

{

string relativePath = file.Substring(sourceDir.Length).TrimStart(Path.DirectorySeparatorChar);

string targetFilePath = Path.Combine(targetDir, relativePath);

Directory.CreateDirectory(Path.GetDirectoryName(targetFilePath));

try

{

File.Copy(file, targetFilePath, true);

int count = Interlocked.Increment(ref copiedCount);

Console.WriteLine($"[INFO] Скопирован: {file} → {targetFilePath} ({count}/{files.Length})");

}

catch (Exception ex)

{

Console.WriteLine($"[ERROR] Ошибка копирования {file}: {ex.Message}");

}

});

Console.WriteLine($"[INFO] Копирование завершено. Всего файлов скопировано: {copiedCount}");

}

}