Лабораторная работа № 4

Разработка клиент-серверных приложений на базе стека протоколов TCP/IP в .Net.

1. Теоретические сведения

Пул потоков в.Net.

Класс ThreadPool обеспечивает приложение пулом рабочих потоков, управляемых системой, позволяя пользователю сосредоточиться на выполнении задач приложения, а не на управлении потоками.Если имеются небольшие задачи, которые нуждаются в фоновой обработки, пул управляемых потоков — это самый простой способ воспользоваться преимуществами нескольких потоков.Например, начиная с.NET Framework 4, можно создавать объекты Task и Task(Of TResult), выполняющие асинхронные задачи в потоках из пула потоков.

Платформа.NET Framework использует потоки из пула потоков в различных целях, включая асинхронное завершение ввода и вывода, обратные вызовы таймера, зарегистрированные операции ожидания, асинхронные вызовы методов с использованием делегатов и подключения сокетов System.Net.

Многие приложение создают потоки, которые большую часть времени находятся в спящем состоянии, ожидая, когда произойдет событие.Другие потоки могут входить в бездействующее состояние только для того, чтобы периодически возобновлять активность для проверки на наличие изменений или обновления сведений о состоянии.

Концепция пула потоков позволяет использовать потоки более эффективно, предоставляя приложению пул рабочих потоков, управляемых системой.Примеры операции, использующие потоки пула потоков :

1. При создании объекта Task или Task(Of TResult) для асинхронного выполнения какой - либо задачи по умолчанию планируется выполнять эту задачу в потоке из пула потоков.
2. Асинхронные таймеры используют пул потоков.Потоки из пула потоков выполняют обратные вызовы из класса System.Threading.Timer и инициируют события из класса System.Timers.Timer.
3. При использовании зарегистрированных дескрипторов ожидания состояние этих дескрипторов ожидания отслеживается системным потоком.Когда операция ожидания завершается, рабочий поток, находящийся в пуле, выполняет соответствующую функцию обратного вызова.

Потоки в управляемом пуле потоков являются фоновыми.То есть их свойства IsBackground имеют значение true.Это означает, что поток ThreadPool не будет

поддерживать выполнение приложения после того, как все потоки переднего плана завершили работу.

Можно также поместить в очередь на обработку пулом потоков рабочие элементы, не связанные с операцией ожидания.Чтобы запросить обработку рабочего элемента потоком из пула, следует вызвать метод QueueUserWorkItem. Этот метод принимает в качестве параметра ссылку на метод или делегат, которые будут вызваны потоком, выбранным из пула потоков.После того, как рабочий элемент помещен в очередь, его обработка не может быть отменена.

Таймеры из очереди таймера и зарегистрированные операции ожидания также используют пул потоков.Их функции обратного вызова помещаются в очередь пула потоков.

Для каждого процесса существует один пул потоков.Начиная с.NET Framework 4, размер по умолчанию пула потоков для процесса зависит от нескольких факторов, таких как размер виртуального адресного пространства.Процесс может вызвать метод GetMaxThreads для определения количества потоков.Количество потоков в пуле потоков может изменяться с помощью метода SetMaxThreads.Каждый поток использует заданные по умолчанию размер стека и приоритет выполнения.

Для увеличения минимального количества потоков можно использовать метод SetMinThreads.Однако необоснованное увеличение этих значений может привести к снижению производительности.Если в одно и то же время запускается слишком много задач, все они могут выполняться слишком медленно.В большинстве случаев пул потоков будет работать лучше, используя свой собственный алгоритм распределения потоков.

Пул потоков по запросу предоставляет новые рабочие потоки или потоки завершения ввода - вывода, пока не будет достигнут минимум для каждой категории.При достижении минимума пул потоков может создавать дополнительные потоки в этой категории или ожидать завершения некоторых задач.Начиная с.NET Framework 4, пул потоков создает и уничтожает рабочие потоки в целях оптимизации пропускной способности, которая определяется как количество задач, завершаемых за единицу времени.Слишком низкое количество потоков может не обеспечить оптимальное использование доступных ресурсов, тогда как слишком большое количество потоков может увеличить количество конфликтов ресурсов.

Когда пул потоков повторно использует поток, он не очищает данные в локальной памяти стека или в полях, помеченных атрибутом ThreadStaticAttribute.Следовательно, данные, помещенные в локальное хранилище стека одним методом, могут быть доступны любому другому методу, выполняющемуся тем же потоком из пула.Метод, который обращается к полю, помеченному атрибутом ThreadStaticAttribute, может обнаружить там различные данные, в зависимости от того, каким именно потоком из пула этот метод выполняется.

Пример пула потоков в .Net

В следующем примере в очередь добавляется задача с помощью метода QueueUserWorkItem, затем методу предоставляются данные задачи.

```
using System;
using System.Threading;
// TaskInfo holds state information for a task that will be
// executed by a ThreadPool thread.
public class TaskInfo
{
        // State information for the task. These members
        // can be implemented as read-only properties, read/write
        // properties with validation, and so on, as required.
        public string Boilerplate;
        public int Value;
        // Public constructor provides an easy way to supply all
        // the information needed for the task.
        public TaskInfo(string text, int number)
        {
        Boilerplate = text;
        Value = number;
        }
}
public class Example
{
        public static void Main()
        {
        // Create an object containing the information needed
        // for the task.
        TaskInfo ti = new TaskInfo("This report displays the number {0}.",
                42);
        // Queue the task and data.
        if (ThreadPool.QueueUserWorkItem(new WaitCallback(ThreadProc), ti))
        {
                Console.WriteLine("Main thread does some work, then sleeps.");
                // If you comment out the Sleep, the main thread exits before
                // the ThreadPool task has a chance to run. ThreadPool uses
                // background threads, which do not keep the application
                // running. (This is a simple example of a race condition.)
                Thread.Sleep(1000);
                Console.WriteLine("Main thread exits.");
        }
        else
        {
                Console.WriteLine("Unable to queue ThreadPool request.");
        }
        }
        // The thread procedure performs the independent task, in this case
        // formatting and printing a very simple report.
```

```
        //
        static void ThreadProc(Object stateInfo)
        {
        TaskInfo ti = (TaskInfo)stateInfo;
        Console.WriteLine(ti.Boilerplate, ti.Value);
        }
}
```

2. Задание на лабораторную работу.

Задания представленные ниже выполняются на технологии .NET.

Варианты заданий

	4-5 баллов
	6-8 баллов
	9-10 баллов

Разработать сервер с использованием пула потоков для параллельной обработки запросов клиентов. К серверу подключается много клиентов, все они отправляют задачи (по вариантам), сервер обрабатывает каждый запрос в отдельном потоке (используя пул потоков) и возвращает ответ клиенту.

Обязательные компоненты отчета:

- Краткое описание алгоритма
- Верификация

Варианты согласуются с преподавателем:

1.	Вычисление псевдослучайного числа используя регистр сдвига с линейной обратной связью (LFSR)
2	Одномерная оптимизация функции методом деления попалам
3	Mergesort
4	Одномерная оптимизация функции методом золотого сечения
5	Quicksort
6	Heapsort
7	Insertion Sort
8	Одномерная оптимизация функции методом равномерного поиска
9	Умножение матриц
10	Одномерная оптимизация функции методом Фибоначчи

11	Treesort
12	Одномерная оптимизация функции методом касательных
13	Разбор математического выражения и вычисление с учетом скобок и приоритетов операций (+ - * /) . Использовать обратную польскую запись
14	Вычисление секанса (без использования математических библиотек, используя разложение в ряд Маклоррена)
15	Вычисление синуса (без использования математических библиотек, используя разложение в ряд Маклоррена)
16	Вычисление экспоненты (без использования математических библиотек, используя разложение в ряд Маклоррена)
17	Разбор математического выражения с учетом приоритетов операций (+ - * /), и тригонометрических функций (sin, cos, tg) . Использовать обратную польскую запись.
18	Вычислить число ПИ до 50 знака методом Монте-Карло
19	Вычислить число ПИ до 50 знака методом иглы Буффона
20	Вычислить число ПИ до 50 знака разложением в ряд Нилаканта
21	Вычислить квадратный корень из числа, используя итерационную формулу Герона
22	Вычислить квадратный корень из числа, используя алгоритм Ньютона
23	Вычисление натурального логарифма (без использования математических библиотек, используя разложение в ряд Маклоррена)
24	Вычисление гиперболического синуса(без использования математических библиотек, используя разложение в ряд Маклоррена)
25	Вычислить число ПИ до 50 знака разложением в ряд Лейбница
26	Решение СЛАУ через обратную матрицу
27	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, строковые константы, заключенные в двойные кавычки и одиночные символы, заключенные в одинарные кавычки. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.

28	Разработать конечный автомат для анализа вырожения. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, римских чисел, знаков операций и скобок. (Римскими считать числа записанные большими буквами X, V и I). Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
29	Разработать конечный автомат для анализа вырожения. Входной язык содержит HTML код, нужно проверить правильность объявления ссылки. Учитывать что могут быть кавычки двух типов или отсутствовать. Проверка атрибутов href, alt, blank. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
30	Разработать конечный автомат для анализа вырожения. Входной язык содержит HTML код, нужно проверить правильность тега img. Учитывать что могут быть кавычки двух типов или отсутствовать. Проверка атрибутов src, alt, width, height. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
31	Разработать конечный автомат для анализа корректности email. Найти все email в файле отправленном пользователем. Клиенту возвращается список всех найденных email в тексте
32	Найти обратную матрицу (используя метод Гаусса)
33	Разбор математического выражения и вычисление с учетом скобок и приоритетов операций (+ - * /) . Использовать дерево выражений
34	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Флойда — Уоршелла
35	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Дейкстры

36	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Беллмана — Форда
37	Решение СЛАУ методом Гаусса
38	Решение СЛАУ методом Жардана-Гауса
39	Найти определитель матрицы
40	LDL^T разложение матрицы
41	Найти обратную матрицу (через единичную)
42	Разложение Холецкого
43	LU разложение матрицы
44	Умножение матриц алгоритмом Штрассена
45	Шифрование строки алгоритмом RSA
46	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка C#, разделенных символом ;(точка с запятой). Считать, что массивы могут содержать только элементы следующих типов: int, double, float, char и String. Массивы инициализируют только фиксированной длины. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
47	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность команд ассемблера в форме: <метка>: <команда> <операнд1>,<операнд2> (метка, а также один или оба операнда могут отсутствовать). В качестве операндов могут выступать регистры процессора 80x86, идентификаторы, целые десятичные числа или целые шестнадцатеричные числа. (Предусмотреть наличие не менее 6 допустимых команд). Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
48	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний записей (record) в соответствии со спецификацией языка Паскаль,

	разделенных символом ;(точка с запятой). Считать, что записи могут содержать только поля скалярных типов integer, real, byte, word, char и строки string с возможным указанием длины строки в квадратных скобках. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
49	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка Паскаль, разделенных символом ;(точка с запятой). Считать, что массивы могут содержать только элементы скалярных типов integer, real, byte, word и char. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
50	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, целые десятичные числа без знака, шестнадцатеричные числа, десятичные числа с плавающей точкой. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
51	Разработать конечный автомат для анализа вырожения. Входной язык содержит выражения над строковыми константами, разделенные символом ;(точка с запятой). Выражения состоят из идентификаторов, строковых констант, заключенных в двойные кавычки, одиночных символов, заключенных в одинарные кавычки и знаков операции конкатенации +. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
52	Разработать конечный автомат для анализа вырожения. Входной язык содержит упрощенные условные операторы типа

	if <логическое выражение> then <оператор присваивания> else <оператор присваивания>; (часть else в операторе может отсутствовать). Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и not. Оператор присваивания должен состоять из двух идентификаторов, разделенных знаком присваивания. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
53	Разработать конечный автомат для анализа вырожения. Входной язык содержит упрощенные операторы цикла типа while <логическое выражение> do <оператор присваивания>; Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и or. Оператор присваивания должен состоять из идентификатора, знака присваивания и целой десятичной константы без знака. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
54	Разработать конечный автомат для анализа вырожения. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знаков операций и скобок. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
55	решения систем обыкновенных дифференциальных уравнений Рунге-Кутты 4-го порядка
56	решения систем обыкновенных дифференциальных уравнений методом Эйлера
57	Разработать конечный автомат для анализа корректности номера телефона в формате.[+XXX (YY) XXX-XX-XX] Найти все номера телефонов в файле отправленном пользователем. Клиенту возвращается список всех найденных номеров телефонов в тексте
58	Разработать конечный автомат для анализа надежности и корректности пароля

	Длина пароля не менее 6 символов. Разрешены только латинские буквы, цифры и знаки: пробел _ - . В пароле обязательно должны присутствовать знаки в верхнем и нижнем регистре и цифры. Пользователю возвращается информация о том, можно ли использовать указанный пароль
59	Медианный фильтр изображения с возможностью выбора размера окна
60	Фильтр Гаусса для изображений с возможностью выбора размера окна
61	Фильтр Собеля (оператор Собеля) для выявления границ объектов изображения
62	Перекрестный оператор Робертса для выявления границ объектов на изображении
63	Оператор Прюитт для выделения границ на изображении
64	Проверка вхождения двух человек в одну группу. Исходные данные, имена людей, которые добавлены в друзья в социальной сети. Пользователь вводит 2 имени, нужно определить связаны ли они через общих друзей. (Find Union)
65	Компрессия/Декомпрессия данных. Алгоритм Лемпеля — Зива — Велча (LZW)
66	Компрессия/Декомпрессия данных. алгоритмом Хаффмана
67	Многомерная оптимизация функции методом деформируемого многогранника (Нелдера-Мида)
68	Многомерная оптимизация функции методом Хука Дживса
69	Многомерная оптимизация функции методом градиентного спуска (построить график)
70	Оператор Кенни для поиска границ объектов на изображении

3. Контрольные вопросы

1. Пул потоков
2. Преимущества и недостатки пула потоков
3. Реализация пула потоков в.Net
4. Использование пула потоков в сетевом приложении для обработки запросов от клиентов