

Работа включает в себя 11 небольших заданий и контрольные вопросы. Не нужно демонстрировать преподавателю каждое приложение. Для защиты работы нужно иметь рабочий программный код и заранее подготовленный отчет со скриншотами. Идеальная защита — до пяти минут. Достаточно продемонстрировать преподавателю умение работать с утилитами (задания 1, 2, 11) и наличие/работоспособность любого приложения по выбору преподавателя. Ответы на вопросы лучше заранее вписать в отчет. Вывод приложений должен быть персонифицирован — как минимум, на скриншоты должно попадать или ваше имя, или название компьютера.

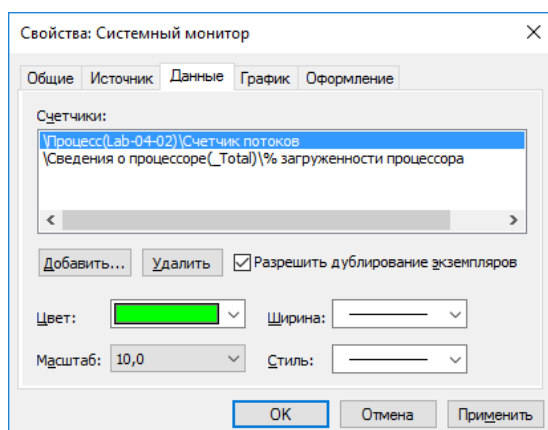
Большая часть требуемого кода приведена в самом задании. Главное – не переписать какой-то код, а понять суть происходящего, для этого активно обсуждайте результаты между собой.

Задание 01

1. Разработайте консольное Windows-приложение **OS04_01** на языке C++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификаторов текущего процесса и текущего потока.
2. Продемонстрируйте информацию об потоках процесса **OS04_01** с помощью утилит **PowerShell ISE** и **Performance Monitor**.

Подсказки

Прежде чем демонстрировать работу преподавателю, заранее запустите **Системный монитор** (perfmon.exe) и добавьте счетчик **Процесс\Счетчик потоков** для экземпляра приложения OS04_01. Масштаб установите 10,0. Для наглядности выберите цвет или воспользуйтесь инструментом **Выделить**.



Командлет **Get-Process** возвращает объект типа **System.Diagnostics.Process**. После запуска процесса сохраните его состояние в переменной, чтобы не «ловить момент», когда приложение еще работает. Можно использовать команду вида:

```
$P = (Get-Process OS04-02)
```

Потом извлеките информацию о потоках из соответствующих свойств. Список свойств и методов можно найти или в документации на .Net Framework, или в самом Powershell с помощью командлета **Get-Member** (алиас **gm**):

```
$p | gm
```

При запуске приложения стартуют еще три потока, которые завершаются примерно через 15 секунд. Их игнорируем.

Задание 02

3. Разработайте консольное Windows-приложение **OS04_02** на языке C++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
4. Процесс **OS04_02** должен создать два потока: потоковые функции **OS04_02_T1**, **OS04_02_T2**.
5. Поток **OS04_02_T1** – выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
6. Поток **OS04_02_T2** – выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
7. Продемонстрируйте информацию об потоках процесса **OS04_02** с помощью утилит **PowerShell ISE** и **Performance Monitor**.

Примечание. Количество итераций можно уменьшить, если использовать подсказки из предыдущего задания. Но 15 секунд все равно придется подождать, чтобы завершились «лишние» потоки.

Лайфхак. Можете добавить свое имя в названия функций или других объектов.

Задание 03.

8. Определите, какое максимальное количество потоков можно создать в одном процессе на вашем компьютере, используя нижеприведенный программный код (или аналогичный). Сравните результаты между собой. От чего зависит

максимальное количество потоков? Какое ограничение накладывает операционная система.

Примечание. Это задание может выполняться больше пяти минут, не тратьте время преподавателя. Просто обсудите результат в группе и впишите в отчет.

```
#include <Windows.h>
#include <iostream>

DWORD WINAPI threadProc(LPVOID);

int main()
{
    HANDLE hThread;
    DWORD dwThreadId;
    for (int Count = 0; Count < 1000000; Count++)
    {
        hThread = CreateThread(
            NULL, 0, threadProc, NULL, CREATE_SUSPENDED, &dwThreadId);
        if (hThread == INVALID_HANDLE_VALUE || hThread == NULL) {
            printf("CreateThread failed (error %d) after %d threads\n",
                GetLastError(), Count);
            break;
        }
        ResumeThread(hThread);
        CloseHandle(hThread);
        if (Count % 1000 == 0)
            printf("%d\n", Count);
    }
    printf("Thread Main exited\n");
    return 0;
}

DWORD WINAPI threadProc(LPVOID args)
{
    Sleep(600000);
    return 0;
}
```

Задание 04

9. Создайте консольное приложение на C#, которое запускает три дополнительных потока и завершается через пять секунд. Один дополнительный поток завершается через 10 секунд, остальные два – через двадцать секунд. Фрагмент программного кода приведен ниже. (Вставьте Свои Имя-Фамилию).

```
class Program
{
    // Поток Z работает 10 секунд
    static void ThreadZed()
    {
        for (int i = 0; i < 10; i++)
        {
            Console.Write(" (Z-{0}) ", Thread.CurrentThread.ManagedThreadId);
        }
    }
}
```

```

        Thread.Sleep(1000);
    }
    Console.WriteLine(" Поток Z завершается ");
}
// Поток работает 20 секунд, параметр - строка-идентификатор
static void ThreadWithParam(object o)
{
    for (int i = 0; i < 20; i++)
    {
        Console.Write(" ({0}-{1}) ", o.ToString(),
            Thread.CurrentThread.ManagedThreadId);
        Thread.Sleep(1000);
    }
}
static void Main(string[] args)
{
    var t1 = new Thread(ThreadZed);
    var t1a = new Thread(ThreadWithParam);
    var t1b = new Thread(ThreadWithParam);
    t1.IsBackground = true;    // false для п.11
    t1a.IsBackground = true;   // false для п.12
    t1b.IsBackground = true;
    t1.Start();
    t1a.Start("Мих"); // Имя
    t1b.Start("Преп"); // Фамилия
    // Главный поток работает 5 секунд
    for (int i = 0; i < 5; i++)
    {
        Console.Write(" (*-{0}) ", Thread.CurrentThread.ManagedThreadId);
        Thread.Sleep(1000);
    }
    Console.WriteLine("Главный поток завершается");
}
}

```

10. Выполните приложение. Обратите внимание на общее время работы приложения. < 📷 >
11. Измените значение свойства **IsBackground** для первого дополнительного потока на **false** и снова выполните приложение. Обратите внимание на общее время работы приложения. < 📷 >
12. Измените значение свойства **IsBackground** для второго дополнительного потока на **false** и снова выполните приложение. Обратите внимание на общее время работы приложения. < 📷 >
13. Вставьте три скриншота в отчет и объясните результат.
14. [For nerds only] Как получить аналогичный результат в «чистом» Windows API?

Задание 05 (вспомогательное)


15. Создайте функцию, которая производит ЛЮБЫЕ вычисления длительностью **n** миллисекунд на вашем компьютере (для последующих заданий метод `Thread.Sleep(n)` не подходит, так как он освобождает центральный процессор и ничего не делает). Убедитесь, что `MySleep(10000)` работает ровно 10 секунд.

Один из вариантов (не забывайте про оптимизатор: он может исключать вычисления, результат которых не используется; в данном примере накапливается сумма и возвращается из метода).

```
static Double MySleep(int ms)
{
    Double Sum = 0, Temp;
    for (int t = 0; t < ms; ++t)
    {
        Temp = 0.711 + (Double)t / 10000.0;
        Double a, b, c, d, e, nt;
        for (int k = 0; k < 5500; ++k)
        {
            nt = Temp - k / 27000.0;
            a = Math.Sin(nt);
            b = Math.Cos(nt);
            c = Math.Cos(nt / 2.0);
            d = Math.Sin(nt / 2);
            e = Math.Abs(1.0 - a * a - b * b) + Math.Abs(1.0 - c * c - d * d);
            Sum += e;
        }
    }
    return Sum;
}
```

16. Узнайте количество ядер и логических процессоров в вашем компьютере (приложение 1).

Задание 06

17. Разработайте консольное приложение **OS04_06** на языке C#, запускающее 20 потоков, каждый из которых в цикле 5000000 раз увеличивает на единицу значение общей для всех потоков переменной. Исходное значение переменной — ноль. Выведите результат и сравните с произведением 20x5000000. <  >

```
class Program
{
    static int Count = 0;

    static void WorkThread()
    {
        for (int i = 0; i < 5000000; ++i)
            Count = Count + 1;
    }
}
```

```

    }


    static void Main(string[] args)
    {
        Thread[] t = new Thread[20];
        for (int i = 0; i < 20; ++i)
        {
            t[i] = new Thread(WorkThread);
            t[i].Start();
        }
        for (int i = 0; i < 20; ++i)
            t[i].Join();
        Console.WriteLine(Count);
        Console.WriteLine(20 * 5000000);
    }
}

```

18. Сравните результаты в группе. Есть ли какая закономерность?

Примечание. В лабораторной работе №6 это приложение нужно будет исправить.

Задание 07

19. Разработайте консольное приложение **OS04_07** на языке C#, запускающее N потоков, каждый из которых будет производить вычисления t секунд (использовать разработанный в задании 5 метод), используя класс System.Threading.Thread. Сохраните информацию о работе потоков в течение T секунд и выведите на экран в виде таблицы <  >. Подберите подходящие параметры в зависимости от количества логических процессоров в вашем компьютере (например, для четырех логических процессоров N = 10, t = 10, T=30).

```

class Program
{
    const int ThreadCount = 10;
    const int ThreadLifeTime = 10;
    const int ObservationTime = 30;
    static int[,] Matrix = new int[ThreadCount, ObservationTime];
    static DateTime StartTime = DateTime.Now;

    static void WorkThread(object o)
    {
        int id = (int)o;
        for (int i = 0; i < ThreadLifeTime * 20; i++)
        {
            DateTime CurrentTime = DateTime.Now;
            int ElapsedSeconds =
                (int)Math.Round(CurrentTime.Subtract(StartTime).TotalSeconds - 0.49);
            Matrix[id, ElapsedSeconds] += 50;
            MySleep(50);           // из задания 5
        }
    }
}

```

```

    }

    static void Main(string[] args)
    {
        Thread[] t = new Thread[ThreadCount];
        for (int i = 0; i < ThreadCount; ++i)
        {
            object o = i;
            t[i] = new Thread(WorkThread);
            t[i].Start(o);
        }
        Console.WriteLine("A student ... is waiting for the threads to finish");
        for (int i = 0; i < ThreadCount; ++i)
            t[i].Join();

        for (int s = 0; s < ObservationTime; s++)
        {
            Console.Write("{0,3}: ", s);
            for (int th = 0; th < ThreadCount; th++)
            {
                Console.Write(" {0,5}", Matrix[th,s]);
            }
            Console.WriteLine();
        }
    }

    static Double MySleep(int ms)
    {
        ...
    }
}

```

Задание 08


20. Скопируйте консольное приложение **OS04_07** как **OS04_08**. Теперь используйте пул потоков. Выведите статистику работы потоков на экран в виде таблицы < 📷 >.

```

    static void Main(string[] args)
    {
        Console.WriteLine("A student ... is placing threads to the pool...");
        for (int i = 0; i < ThreadCount; ++i)
        {
            object o = i;
            ThreadPool.QueueUserWorkItem(WorkThread, o);
        }
        Console.WriteLine("A student ... is waiting for the threads to finish...");
        Thread.Sleep(1000 * ObservationTime);
        for (int s = 0; s < ObservationTime; s++)
        {
            Console.Write("{0,3}: ", s);
            for (int th = 0; th < ThreadCount; th++)
            {
                Console.Write(" {0,5}", Matrix[th, s]);
            }
            Console.WriteLine();
        }
    }
}

```


Задание 09

21. Скопируйте консольное приложение **OS04_07** как **OS04_09**. На этот раз используйте `System.Threading.Tasks.Task`. Выведите статистику работы потоков на экран в виде таблицы <  >. Сравните результаты заданий 7-9 и запишите вывод в отчет.

```
static void Main(string[] args)
{
    Task[] t = new Task[TaskCount];
    int[] numbers = new int[TaskCount];
    for (int i = 0; i < TaskCount; i++)
        numbers[i] = i;
    Console.WriteLine("A student ... is creating tasks...");
    t[0] = Task.Run(() => { Work(0); });
    t[1] = Task.Run(() => { Work(1); });
    t[2] = Task.Run(() => { Work(2); });
    t[3] = Task.Run(() => { Work(3); });
    t[4] = Task.Run(() => { Work(4); });
    t[5] = Task.Run(() => { Work(5); });
    t[6] = Task.Run(() => { Work(6); });
    t[7] = Task.Run(() => { Work(7); });
    t[8] = Task.Run(() => { Work(8); });
    t[9] = Task.Run(() => { Work(9); });
    Console.WriteLine("A student ... is waiting for tasks to finish...");
    Task.WaitAll(t);
    for (int s = 0; s < ObservationTime; s++)
    {
        Console.Write("{0,3}: ", s);
        for (int th = 0; th < TaskCount; th++)
            Console.Write(" {0,5}", Matrix[th, s]);
        Console.WriteLine();
    }
}
```

22. [For nerds only] Реализуйте запуск задач в цикле.

Задание 10

23. Скопируйте консольное приложение **OS04_09** как **OS04_10**. Уменьшите количество задач до количества логических процессоров. Организуйте выполнение задач по очереди. Выведите статистику работы потоков на экран в виде таблицы <  >.

```
static void Main(string[] args)
{
    Task[] t = new Task[TaskCount];
    int[] numbers = new int[TaskCount];
    for (int i = 0; i < TaskCount; i++)
        numbers[i] = i;
    Console.WriteLine("A student ... is creating tasks...");
    t[0] = new Task(() => { Work(0); });
    t[0].Start();
    t[1] = t[0].ContinueWith(delegate { Work(1); });
    t[2] = new Task(() => { Work(2); });
    t[2].Start();
    t[3] = t[1].ContinueWith(delegate { Work(3); });
}
```



```

        Console.WriteLine("A student ... is waiting for tasks to finish...");
        Task.WaitAll(t);
        for (int s = 0; s < ObservationTime; s++)
        {
            Console.Write("{0,3}: ", s);
            for (int th = 0; th < TaskCount; th++)
                Console.Write(" {0,5}", Matrix[th, s]);
            Console.WriteLine();
        }
    }
}

```

Задание 11

24. Разработайте на языке консольное Linux-приложение **OS04_10** на языке C, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса (использовать функции из pthread.h).
25. Процесс **OS04_10** должен создать поток: потоковая функция **OS04_10_T1**.
26. Поток **OS04_10_T1** - выполняет цикл 75 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса.
27. Продемонстрируйте информацию о потоках процесса **OS04_10** с помощью утилиты **ps**.

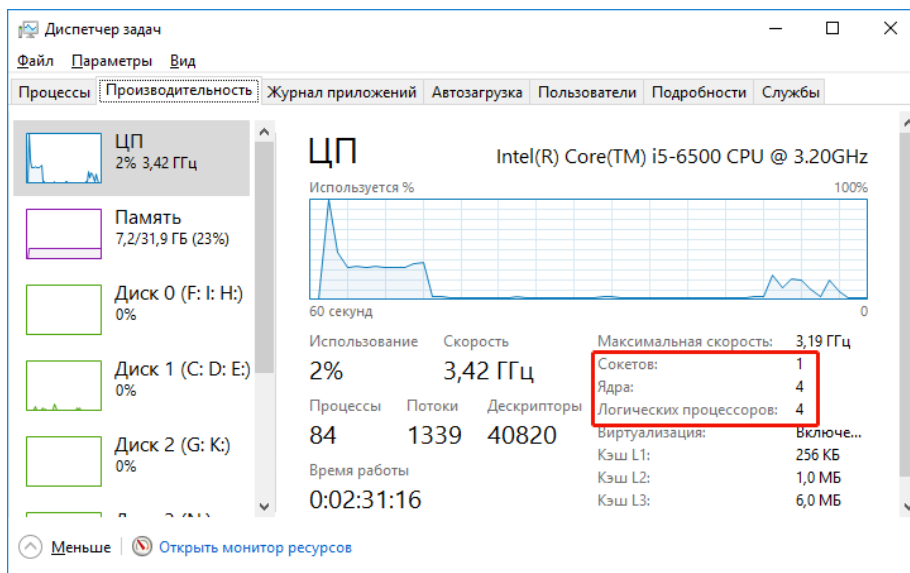
Задание 12. Ответьте на следующие вопросы

28. Что такое поток управления OS?
29. С помощью каких системных вызовов создаются потоки в Windows и Linux?
30. Что такое системные и пользовательские потоки?
31. Что такое многопоточность?
32. Что такое контекст потока и для чего он нужен?
33. Перечислите состояния, в которых может быть поток и поясните их назначение.
34. Что такое LWP?
35. Что такое потокобезопасность программного кода?
36. Что такое реентерабельность кода?
37. Что такое Fiber?

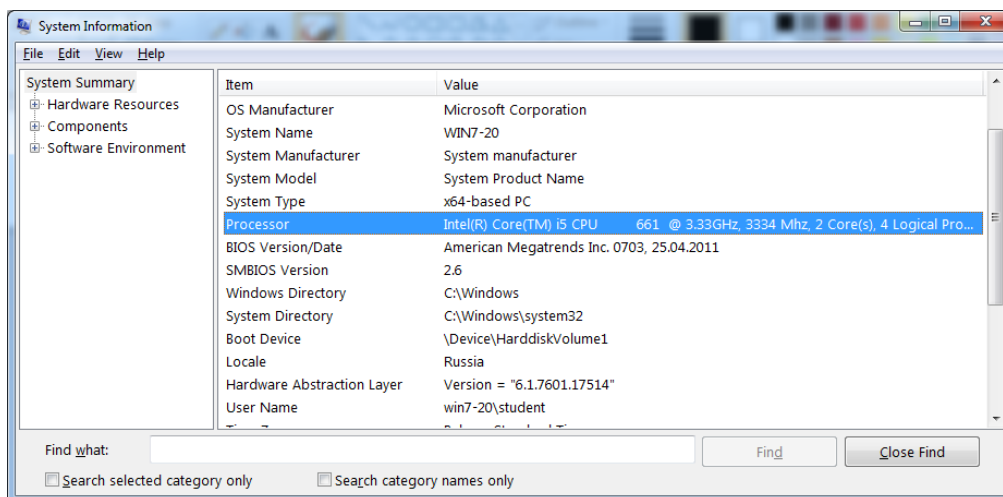
Приложение 1. Определение количества логических процессоров

Запустите Диспетчер задач (Task Manager), например, нажав комбинацию клавиш **Ctrl+Shift+Esc**. Перейдите на закладку **Производительность** и посмотрите, сколько в вашем компьютере имеется процессорных ядер и логических процессоров.

В современных компьютерах установлены, как правило, многоядерные процессоры. В примере ниже процессор имеет четыре ядра.

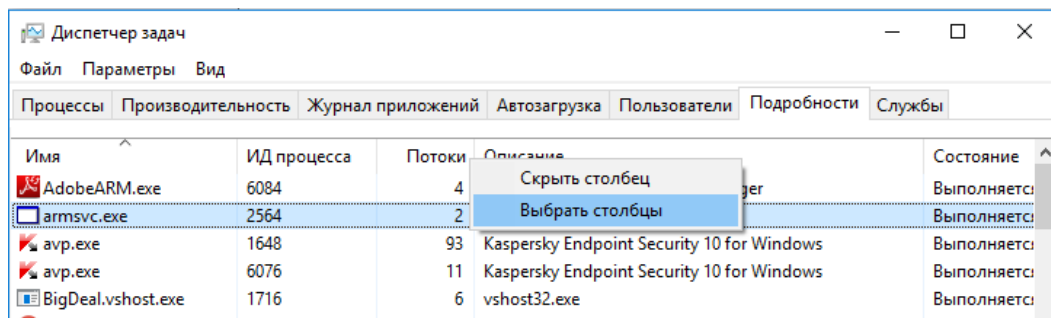


Альтернатива — запустить программу `msinfo32.exe`.



Если никогда ранее не использовали Диспетчер задач, ознакомьтесь с программой, зайдя на каждую вкладку.

Настройте вкладку **Подробности**, чтобы она отображала количество потоков в каждом процессе (щелкните правой кнопкой на строке заголовка списка процессов и выберите команду **Выбрать столбцы**).



В открывшемся диалоговом окне поставьте птичку напротив варианта **Потоки** и нажмите кнопку **ОК**. Затем можно перетащить столбец **Потоки** ближе к левому краю таблицы.

