

Для наглядности результата, в заданиях 2 и 3 нужно подобрать количество потоков и используемых логических процессоров.

Как **минимум**, при защите работы нужно:

- 1) наличие рабочего программного кода для всех заданий;
- 2) отчет о работе с персонифицированными скриншотами (например, фамилия внутри скриншота), выводами и краткими ответами на вопросы;
- 3) лист бумаги с фамилией и нарисованными от руки схемами приоритетов в Windows и Linux и подписанными числовыми значениями (пояснить ассистенту, как вычисляется базовый приоритет потока в Windows и связь между niceness и приоритетом в Linux).

**Дополнительные требования** могут устанавливаться ассистентом (включая, но не ограничиваясь: запуск программы на выбор, ответ на контрольный вопрос по выбору и т. д. и т. п.).

### **Задание 01**

1. Разработайте консольное Windows-приложение **OS05\_01** на языке C++, выводящее на консоль следующую информации:
  - идентификатор текущего процесса;
  - идентификатор текущего (main) потока;
  - приоритет (приоритетный класс) текущего процесса;
  - приоритет текущего потока;
  - маску (affinity mask) доступных процессу процессоров в двоичном виде;
  - количество процессоров, доступных процессу;
  - процессор, назначенный текущему потоку.

### **Задание 02**

2. Создайте консольное Windows **OS05\_02** на языке C#, взяв за основу приложение **OS04\_07** из Лабораторной работы №4. Измените метод **Main** таким образом, чтобы потоки 0, 3, 6 и т.д. запускались с минимальным приоритетом потока, а потоки 2, 5, 8... – с максимальным. Класс приоритета процесса оставьте по умолчанию (Normal). Как пример, можно использовать следующий фрагмент кода.

```

for (int i = 0; i < ThreadCount; ++i)
{
    object o = i;
    t[i] = new Thread(WorkThread);
    switch (i % 3)
    {
        case 0:
            t[i].Priority = ThreadPriority.Lowest;
            break;
        case 2:
            t[i].Priority = ThreadPriority.Highest;
            break;
    }
    t[i].Start(o);
}

```


Примечание. Отношение количества потоков к количеству логических процессоров должно быть строго больше двух, иначе требуемый эффект не проявится. Если на вашем компьютере количество логических процессоров больше шести, результат может плохо читаться из-за переносов строк. Можете принудительно ограничить количество используемых логических процессоров (маска 15 для четырех, 31 для пяти и т.д.), задав маску в начале метода Main:

```

// Если у вас слишком много логических процессоров - ограничьте
// количество используемых процессом. Внимание - здесь битовая маска!
Process.GetCurrentProcess().ProcessorAffinity = (System.IntPtr)15;

```

И метод MySleep должен работать именно 1 миллисекунду, нужно адаптировать его под свой процессор. Для процессоров с холодными и горячими ядрами ориентируйтесь на горячие.

3. Выполните приложение <  >, не забудьте про персонификацию вывода результатов.
4. По зафиксированным скриншотам объясните полученные результаты.

### Задание 03

5. Создайте консольное Windows **OS05\_03** на языке C#, взяв за основу приложение **OS05\_02** из настоящей работы. На этот раз только несколько потоков запустите на наименьшем приоритете потока, а остальные – на наибольшем.

```

for (int i = 0; i < ThreadCount; ++i)
{
    object o = i;
    t[i] = new Thread(WorkThread);
    if (i < 2) // здесь 2 - половина логических процессоров
        t[i].Priority = ThreadPriority.Lowest;
    else
        t[i].Priority = ThreadPriority.Highest;
    t[i].Start(o);
}

```

6. Выполните приложение **OS05\_03** < 📷 >. Удалось ли поработать низкоприоритетным потокам? (Чтобы уменьшить влияние случайности, можно повторить эксперимент несколько раз).
7. Выполните приложение **OS05\_03** с другими парами приоритетов, например, **BelowNormal** и **Normal** < 📷 >. Изменился ли характер работы потоков?
8. По зафиксированным скриншотам объясните полученные результаты. При этом укажите числовые значения приоритетов потоков.

#### **Задание 04**

9. Разработайте консольное Linux-приложение **OS05\_04** на языке C++, выводящее на консоль следующую информации:
  - идентификатор текущего процесса;
  - идентификатор текущего (main) потока;
  - приоритет (nice) текущего потока;
  - номера доступных процессоров.

#### **Задание 05**

10. Разработайте консольное Linux-приложение **OS05\_05** на языке C, выполняющее длинный цикл.
11. Запустите приложение **OS05\_05**.
12. Зафиксируйте < 📷 > текущее значение **nice**, полученное с помощью команды **top**.
13. Увеличьте приоритет для **OS05\_05** до максимального значения (самого привилегированного). Зафиксируйте < 📷 > текущее значение **nice**, полученное с помощью команды **top**.
14. Уменьшите приоритет для **OS05\_05** до минимального значения (самого ничтожного). Зафиксируйте < 📷 > текущее значение **nice**, полученное с помощью команды **top**.

#### **Задание 06.** Ответьте на следующие вопросы

15. Поясните понятие «мультизадачная OS с вытеснением».
16. Поясните понятие «циклическое планирование».

17. Поясните понятие «приоритетное планирование».
18. Поясните понятие «кооперативное планирование».
19. Поясните понятие «OS реального времени».
20. Поясните понятие «приоритет процесса».
21. Поясните выражение «поток уступает процессор другому потоку».
22. Windows: как поток может уступить процессор?
23. Windows: что такое базовый приоритет потока, как он вычисляется и диапазон его изменения?
24. Windows: поясните назначение и принцип применения системного вызова **ResumeThread**.
25. Windows: поясните назначение и принцип применения системного вызова **WaitForSingleObject**.
26. Windows: поясните назначение и принцип применения системных вызовов **GetProcessPriorityBoost**, **GetThreadPriorityBoost**, **SetProcessPriorityBoost**, **SetThreadPriorityBoost**.
27. Linux: поясните принцип идентификации процессов и потоков и поясните, почему он такой.
28. Linux: Поясните понятие «планировщик потоков».
29. Linux: поясните принцип использования значения **nice** – процесса, диапазон его изменения, для какого режима работы планировщика это значение применяется?
30. Linux: перечислите политики планирования, какая действует по умолчанию?
31. Linux: с помощью какого системного вызова поток может уступить процессор.