Лабораторная работа 04

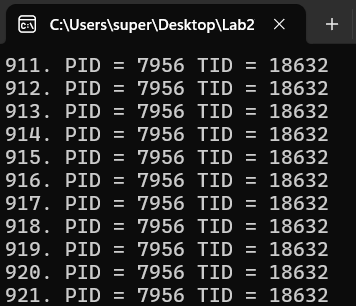
OC, ПОИТ-3, 4 час. (Exp: 20.10.2024)\*

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.

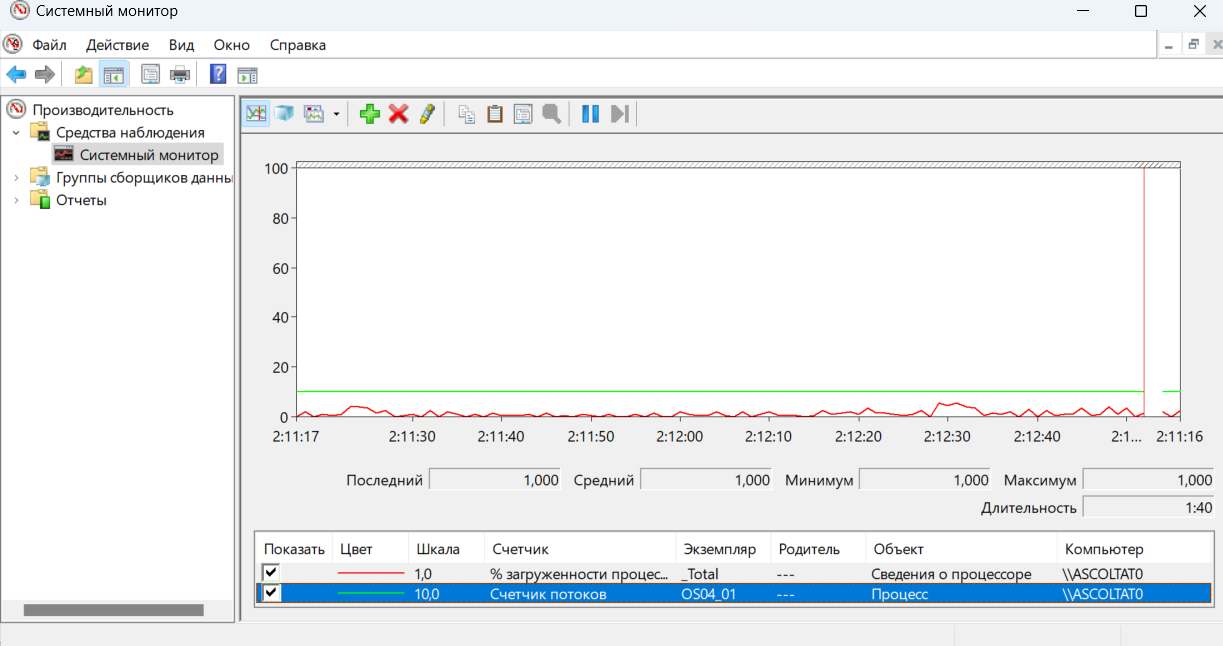
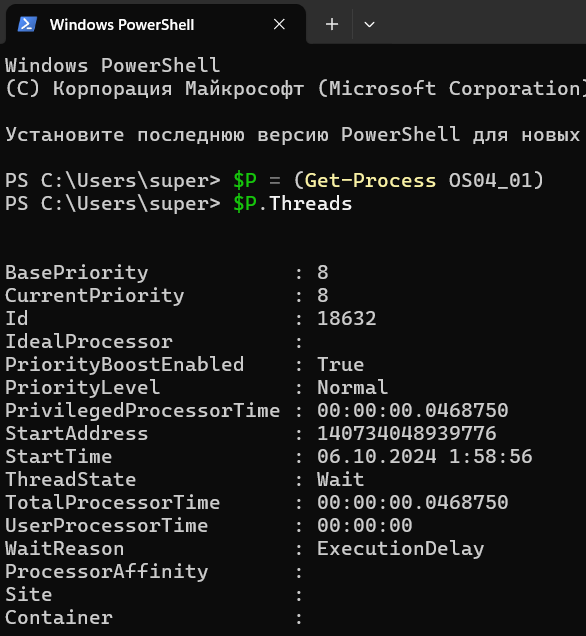
Бо́льшая часть требуемого кода приведена в самом задании. Главное – не переписать какой-то код, а понять суть происходящего, для этого активно обсуждайте результаты между собой.

**Задание 01**

1. Разработайте консольное Windows-приложение **OS04\_01** на языке С++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификаторов текущего процесса и текущего потока.

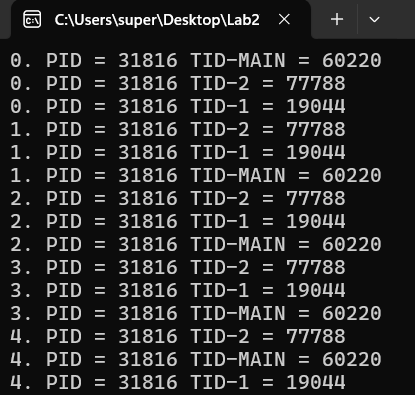


1. Продемонстрируйте информацию об потоках процесса **OS04\_01** с помощью утилит  **PowerShell ISE** и **Performance Monitor**.



**Задание 02**

1. Разработайте на языке консольное Windows-приложение **OS04\_02** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
2. Процесс **OS04\_02** должен создать два потока: потоковые функции **OS04\_02\_T1, OS04\_02\_T2.**
3. Поток **OS04\_02\_T1** - выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
4. Поток **OS04\_02\_T2** - выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.



1. Продемонстрируйте информацию об потоках процесса **OS04\_02** с помощью утилит  **PowerShell ISE** и **Performance Monitor**. Изображение выглядит как текст, снимок экрана, Шрифт, меню

   Автоматически созданное описание Изображение выглядит как текст, снимок экрана, меню, Шрифт

   Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

**Задание 03.**

1. 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.

Максимальное количество потоков зависит от размера выделенного стека для каждого потока, доступной оперативной памяти, архитектуры системы. На архитектуре х32 количество потоков сильно ограничено (около 2000-4000 потоков), а на х64 их может быть десятки тысяч, так как доступно больше виртуальной памяти. На х32 каждый поток занимает определенное количество виртуальной памяти для своего стека, и максимальный размер виртуальной памяти ограничен 2-4 ГБ. По умолчанию размер стека потока в Windows — 1 МБ, но это значение можно изменить при создании потока. Ограничения также связаны с доступными ресурсами процессора, оперативной памяти.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

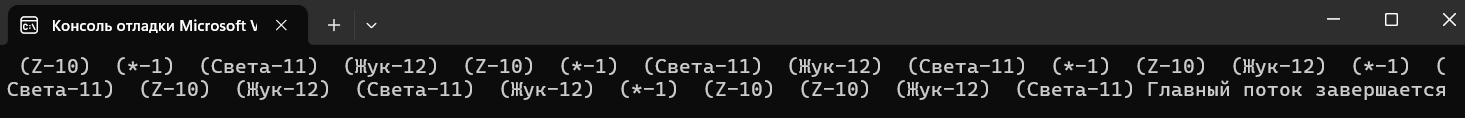
Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

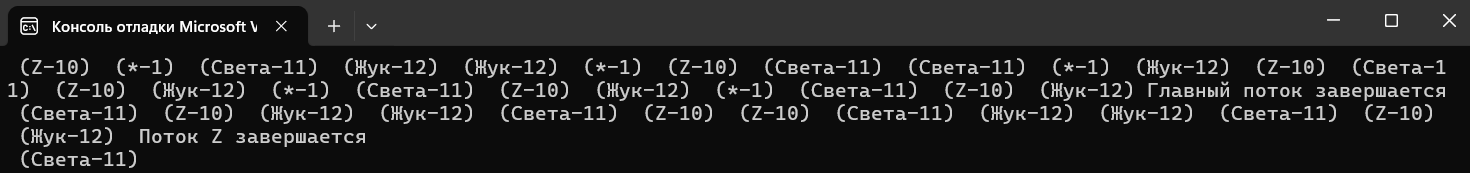
**Задание 04**

1. 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 ИмяФамилию).
2. Выполните приложение. Обратите внимание на общее время работы приложения. < >



5 сек

1. Измените значение свойства IsBackground для первого дополнительного потока на false и снова выполните приложение. Обратите внимание на общее время работы приложения. < >



10 сек

1. Измените значение свойства IsBackground для второго дополнительного потока на false и снова выполните приложение. Обратите внимание на общее время работы приложения. < >

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

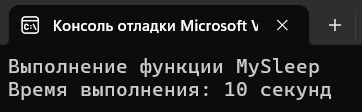
20 сек

1. Вставьте три скриншота в отчет и объясните результат.

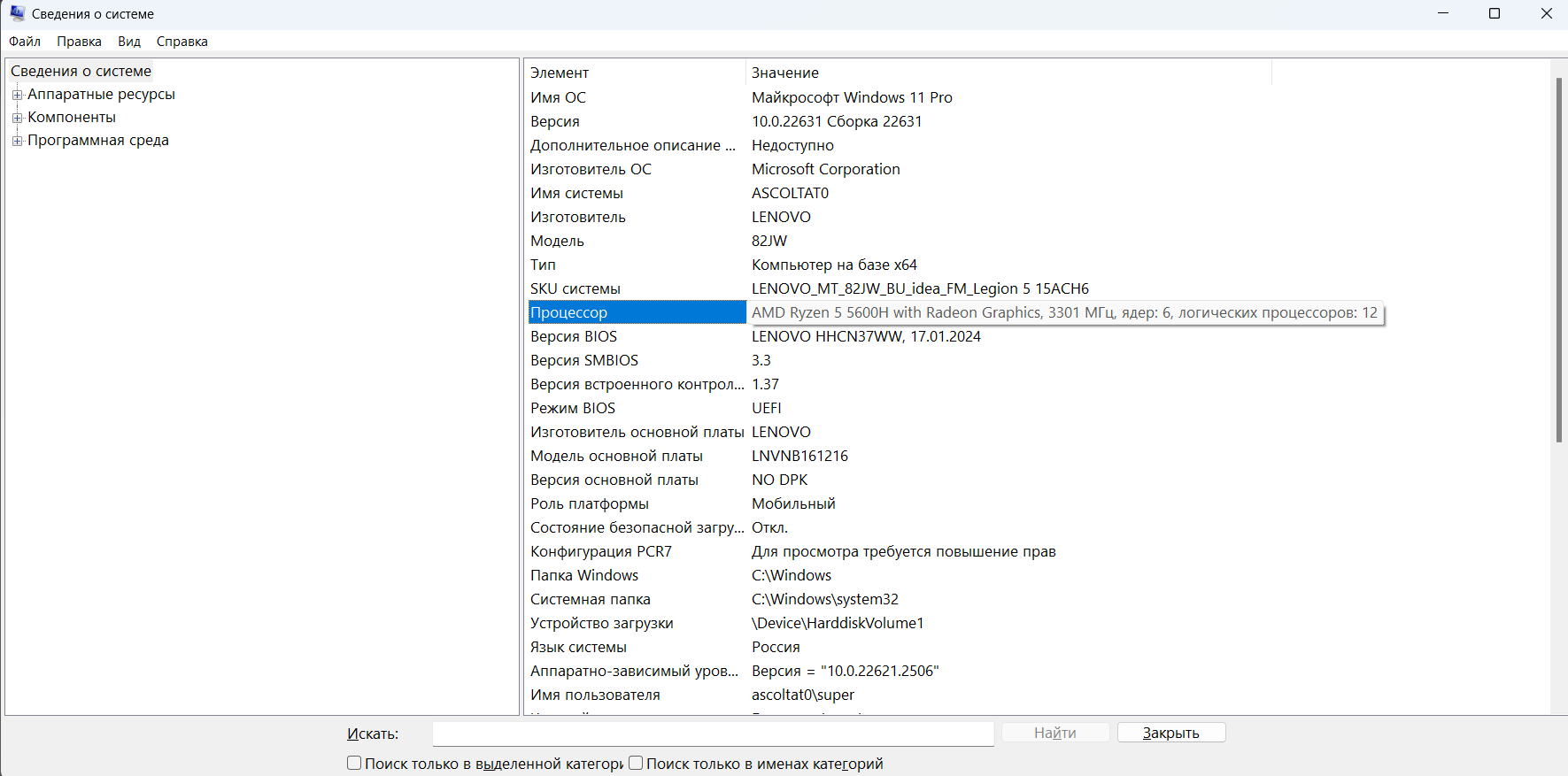
Если IsBackground установлен в true, то основной поток завершится через 5 секунд, и фоновый поток остановится. Если IsBackground установлен в false, фоновый поток продолжит работу даже после завершения основного потока.

**Задание 05**

1. 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.



1. Узнайте количество ядер и логических процессоров в вашем компьютере (приложение 1). (msinfo32.exe)

****

**Задание 06**

1. Разработайте консольное приложение OS04\_06 на языке С#, 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 20х5000000. < >

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

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

Возникает гонка данных при одновременном доступе к одной и той же переменной, поэтому нужно использовать синхронизацию (например, lock), чтобы несколько потоков не могли одновременно изменить значение переменной.

**Задание 07**

1. Разработайте консольное приложение OS04\_07 на языке С#, 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).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Задание 08**

1. Скопируйте консольное приложение OS04\_07 как OS04\_08. Теперь используйте пул потоков. Выведите статистику работы потоков на экран в виде таблицы < >.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Задание 09**

1. 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.

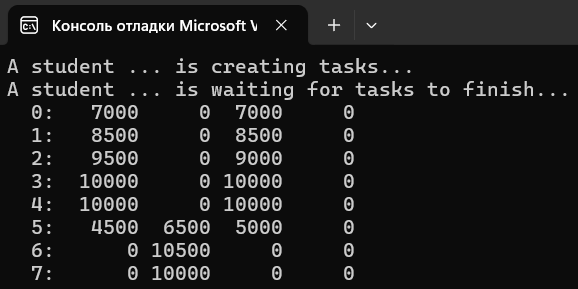
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

В приложении OS04\_08 и OS04\_09 лучше производительность по сравнению с OS04\_07, особенно при увеличении числа потоков, благодаря более эффективному управлению ресурсами и меньшим накладным расходам.

**Задание 10**

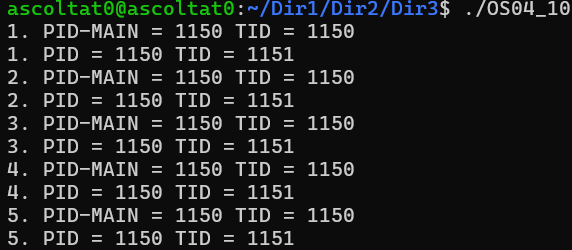
1. Скопируйте консольное приложение OS04\_09 как OS04\_10. Уменьшите количество задач до количества логических процессоров. Организуйте выполнение задач по очереди. Выведите статистику работы потоков на экран в виде таблицы < >.



**Задание 11**

1. Разработайте на языке консольное Linux-приложение OS04\_10 на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса (использовать функции из pthread.h).
2. Процесс OS04\_10 должен создать поток: потоковая функция OS04\_10\_T1.
3. Поток OS04\_10\_T1 - выполняет цикл 75 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса.

|  |
| --- |
| 1. #include <stdio.h> 2. #include <stdlib.h> 3. #include <unistd.h> 4. #include <pthread.h> 5. #include <sys/syscall.h> 6. void\* OS04\_10\_T1(void\* arg) { 7. for (int i = 1; i <= 75; ++i) { 8. printf("%d. PID = %d TID = %lu\n", i, getpid(), syscall(SYS\_gettid)); 9. sleep(1); 10. } 11. } 12. int main() { 13. pthread\_t thread; 14. pthread\_create(&thread, NULL, OS04\_10\_T1, NULL); 15. for (int i = 1; i <= 100; ++i) { 16. printf("%d. PID-MAIN = %d TID = %lu\n", i, getpid(), syscall(SYS\_gettid)); 17. sleep(1); 18. }   } |



1. Продемонстрируйте информацию о потоках процесса OS04\_10 с помощью утилиты ps.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

**Задание 12.ответьте на следующие вопросы**

1. Что такое поток управления OS?

Поток управления(OS) – наименьшая единица работы ядра ОС, являющаяся частью процесса.

1. С помощью каких системных вызовов создаются потоки в Windows и Linux?

- В Windows CreateThread().

- В Linux pthread\_create().

1. Что такое системные и пользовательские потоки?

Системные потоки - потоки, управляемые ядром ОС. Пользовательские потоки - потоки, управляемые самим приложением (не взаимодействует напрямую с ядром ОС).

1. Что такое многопоточность?

Многопоточность – способность ЦП исполнять несколько потоков одновременно в пределах одного процесса. Упрощает разделение задач и реализацию параллельных вычислений.

1. Что такое контекст потока и для чего он нужен?

Контекст потока - набор данных, необходимых для восстановления работы потока при его остановке: программный код, набор регистров, стек памяти и ядра ОС, маркер доступа.

1. Перечислите состояния в которых может быть поток и поясните их назначение.

- Создан (New): поток создан.

- Готовый (Ready): поток готов к выполнению, но ожидает своей очереди.

- Запущенный (Running): поток выполняется.

- Заснул на некоторое время (Sleeping): заснул на время.

- Заблокированный (Blocked/Waiting): заблокировать до наступления события.

-Завершенный (Terminated/Finish): поток успешно завершился или был принудительно остановлен.

1. Что такое LWP?

LWP (Lightweight Process) - легковесный процесс, который является базовым элементом многопоточной реализации в Linux.

1. Что такое потокобезопасность программного кода?

Значит, что код может быть безопасно выполнен одновременно несколькими потоками. Это достигается с помощью синхронизации (мьютексы, семафоры, критические секции).

1. Что такое реентерабельность кода?

Реентерабельность кода - свойство программного кода, которое позволяет ему быть корректно выполненным в ситуации, когда один и тот же код вызывается из нескольких потоков одновременно.

1. Что такое Fiber?

Fiber - легковесный поток выполнения, который контролируется самим приложением, а не ОС. Нужен для ручного планирования выполнения кода в рамках потока. Находится внутри потоков (процессы –> потоки –> волокна).