Лабораторная работа 05

OC, ПОИТ-3

**Задание 01**

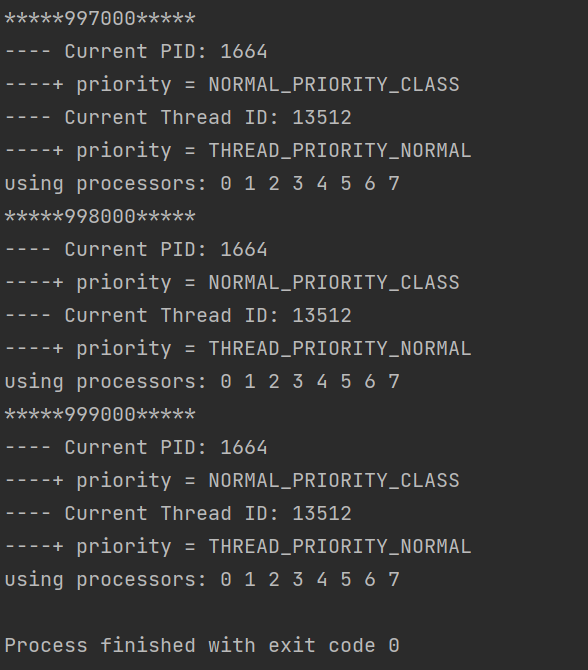
1. Разработайте консольное Windows-приложение **OS05\_01** на языке С++, выводящее на консоль следующую информации:

* 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**

1. Разработайте консольное Windows-приложение **OS05\_02x,** выполняющее цикл в 1млн итераций.
2. Каждая итерация осуществляет задержку на 200 мс через каждые 1тыс итераций и выводит следующую информацию:

* номер итерации;
* идентификатор процесса;
* идентификатор потока;
* класс приоритета процесса;
* приоритет потока:
* номер назначенного процессора.



1. Разработайте консольное Windows-приложение **OS05\_02,** принимающее следующие параметры:

* P1: целое число, задающее маску доступности процессоров (affinity mask);
* P2: целое число, задающее класс приоритета первого дочернего процесса;
* P3: целое число, задающее класс приоритета второго дочернего процесса.

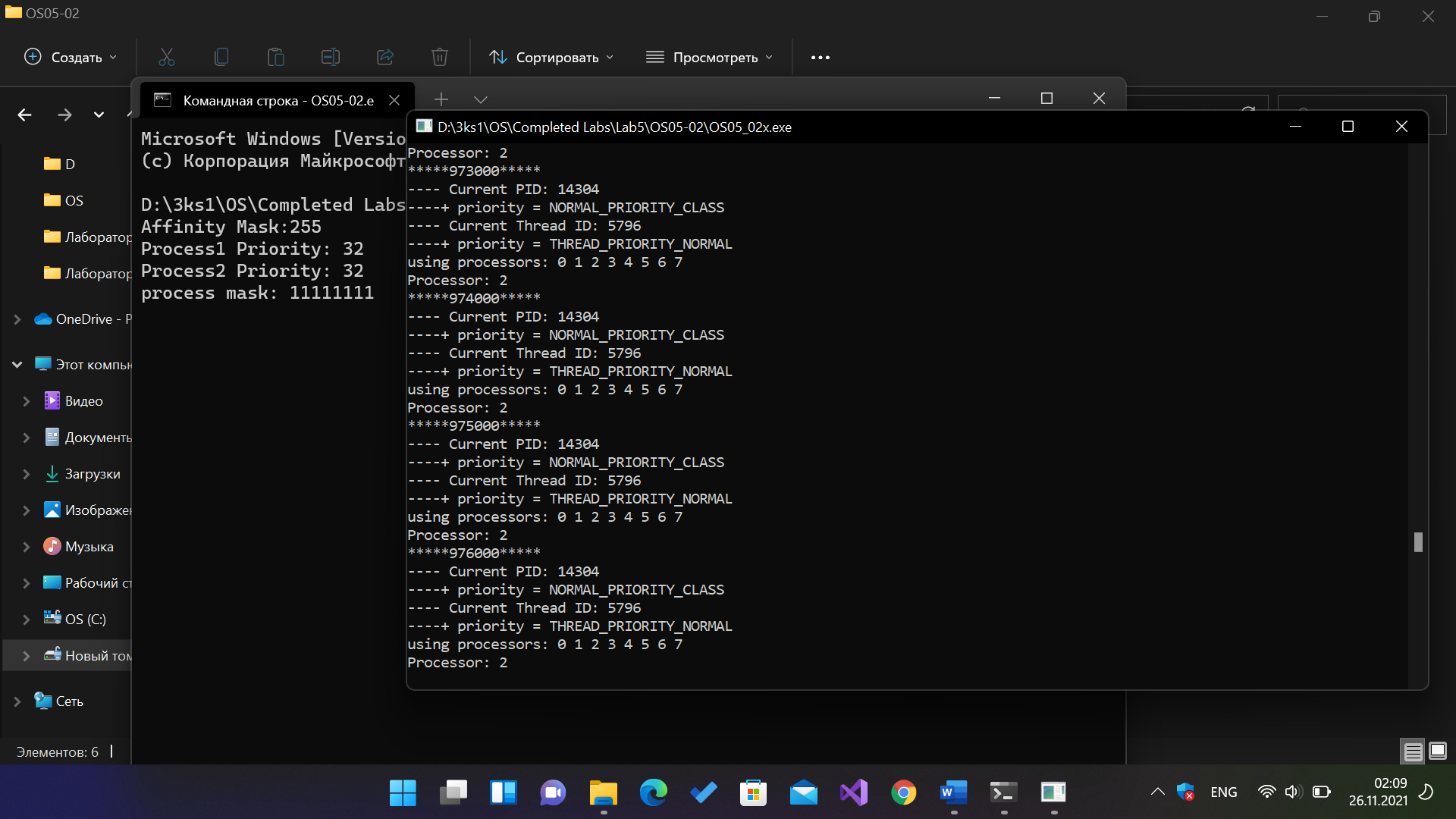
1. Приложение **OS05\_02** должно вывести в свое консольное окно заданные параметры и запустить два одинаковых дочерних процесса **OS05\_02x,** осуществляющих вывод в отдельные консольные окна и имеющих заданные в параметрах приоритеты.
2. Запустите приложение **OS05\_02**,принимающее следующие значения параметров:

* P1: доступны все процессоры;
* P2: Normal;
* P3: Normal.

(255 32 32)

(255 32 32)

Зафиксируйте (скриншот) в момент первого окончания одного из дочерних процессов расхождение в количестве выполненных процессами итераций.



1. Запустите приложение **OS05\_02**,принимающее следующие значения параметров:

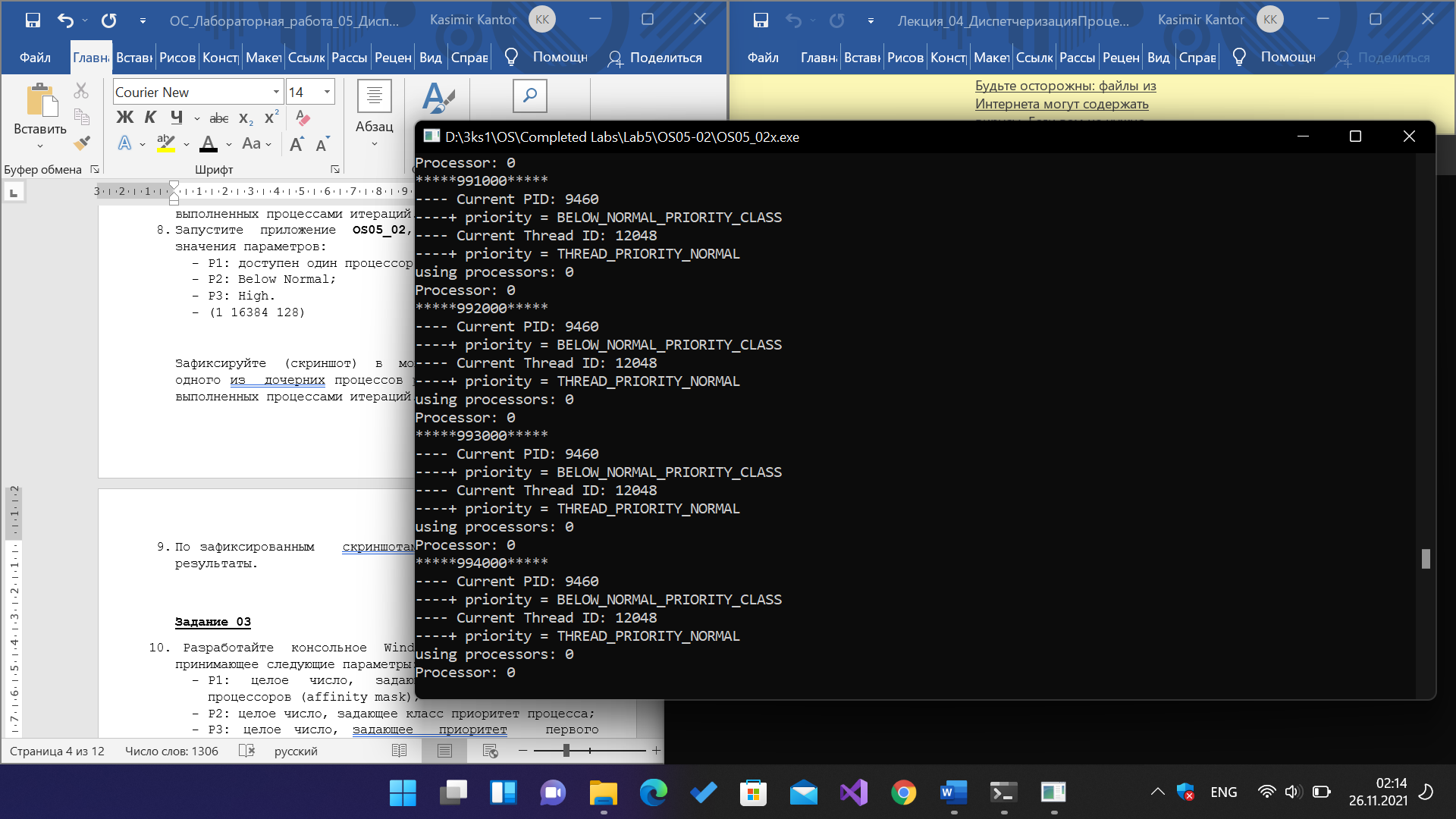
* P1: доступны все процессоры;
* P2: Below Normal;
* P3: High.

(255 16384 128)

Зафиксируйте (скриншот) в момент первого окончания одного из дочерних процессов расхождение в количестве выполненных процессами итераций.

1. Запустите приложение **OS05\_02**,принимающее следующие значения параметров:

* P1: доступен один процессор;
* P2: Below Normal;
* P3: High.
* (1 16384 128)

Зафиксируйте (скриншот) в момент первого окончания одного из дочерних процессов расхождение в количестве выполненных процессами итераций. 

1. По зафиксированным скриншотам, объясните полученные результаты.

**Задание 03**

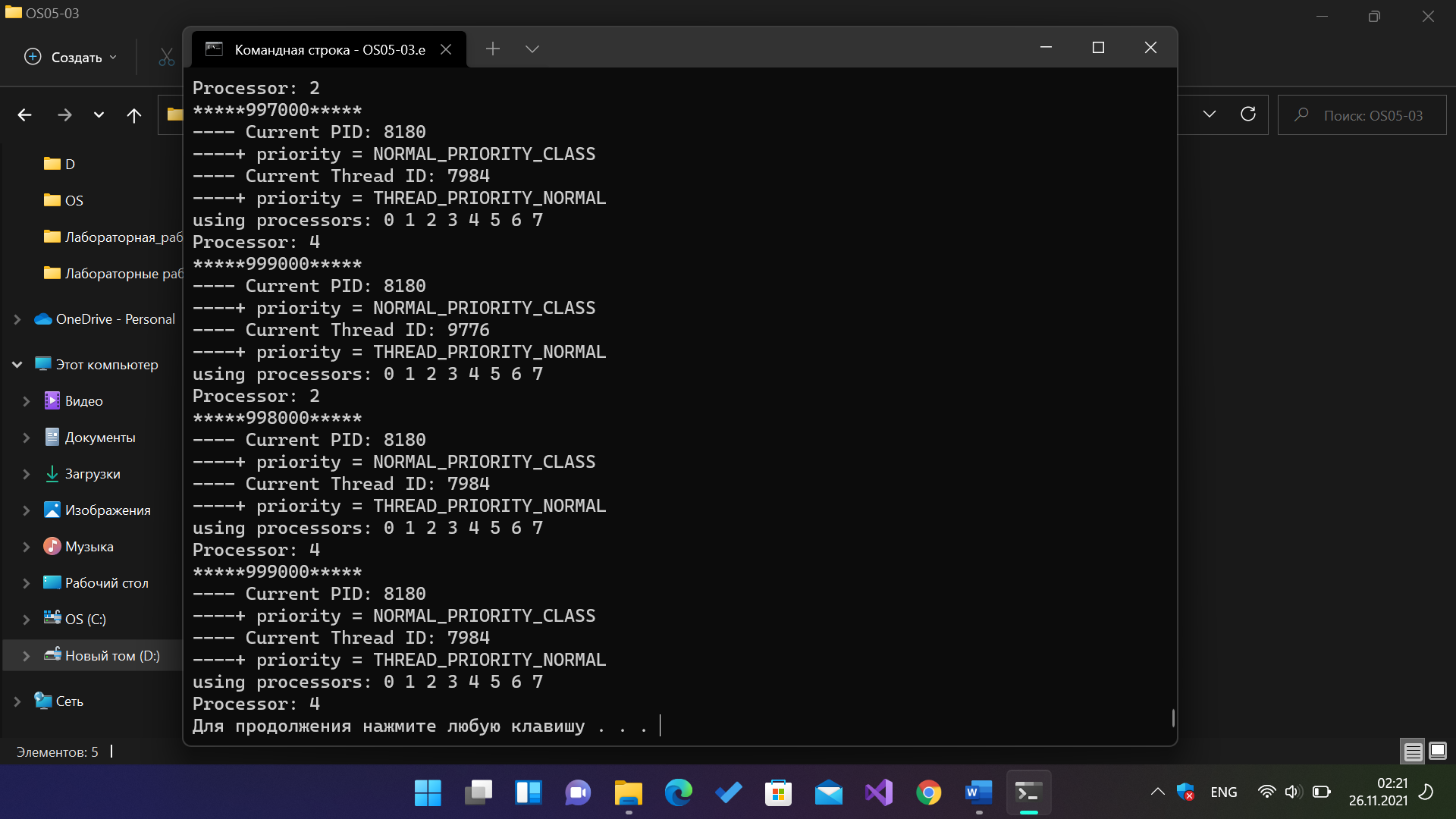
1. Разработайте консольное Windows-приложение **OS05\_03,** принимающее следующие параметры:

* P1: целое число, задающее маску доступности процессоров (affinity mask);
* P2: целое число, задающее класс приоритет процесса;
* P3: целое число, задающее приоритет первого дочернего потока;
* P4: целое число, задающее приоритет второго дочернего потока.

1. Приложение **OS05\_03**  включает в себя потоковую функцию **TA**, выполняющую цикл в 1млн итераций, аналогичный циклу в задании 02.
2. Приложение **OS05\_03**  должно вывести в свое консольное окно заданные параметры и запустить два одинаковых дочерних потока (потоковая функция **TA**)**,** осуществляющих вывод консольное окно и имеющих заданные в параметрах приоритеты.
3. Запустите приложение **OS05\_03**,принимающее следующие значения параметров:

* P1: доступны все процессоры;
* P2: Normal;
* P3: Normal;
* P4: Normal;

(255 32 0 0)

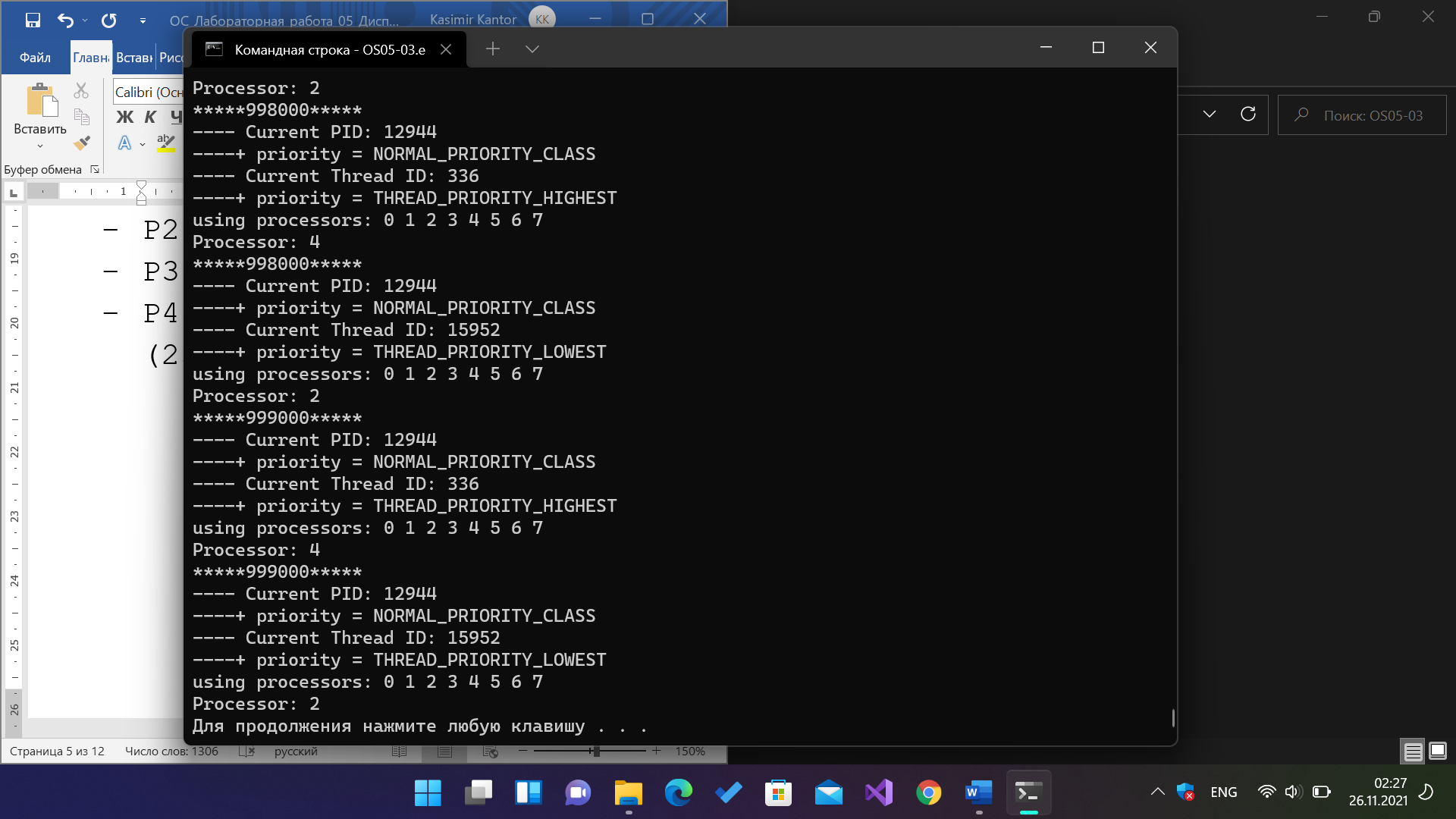


Зафиксируйте (скриншот) в момент первого окончания одного из дочерних потоков расхождение в количестве выполненных потоками итераций.

1. Запустите приложение **OS05\_03**,принимающее следующие значения параметров:

* P1: доступны все процессоры;
* P2: Normal;
* P3: Lowest;
* P4: Highest.

(255 32 -2 2)

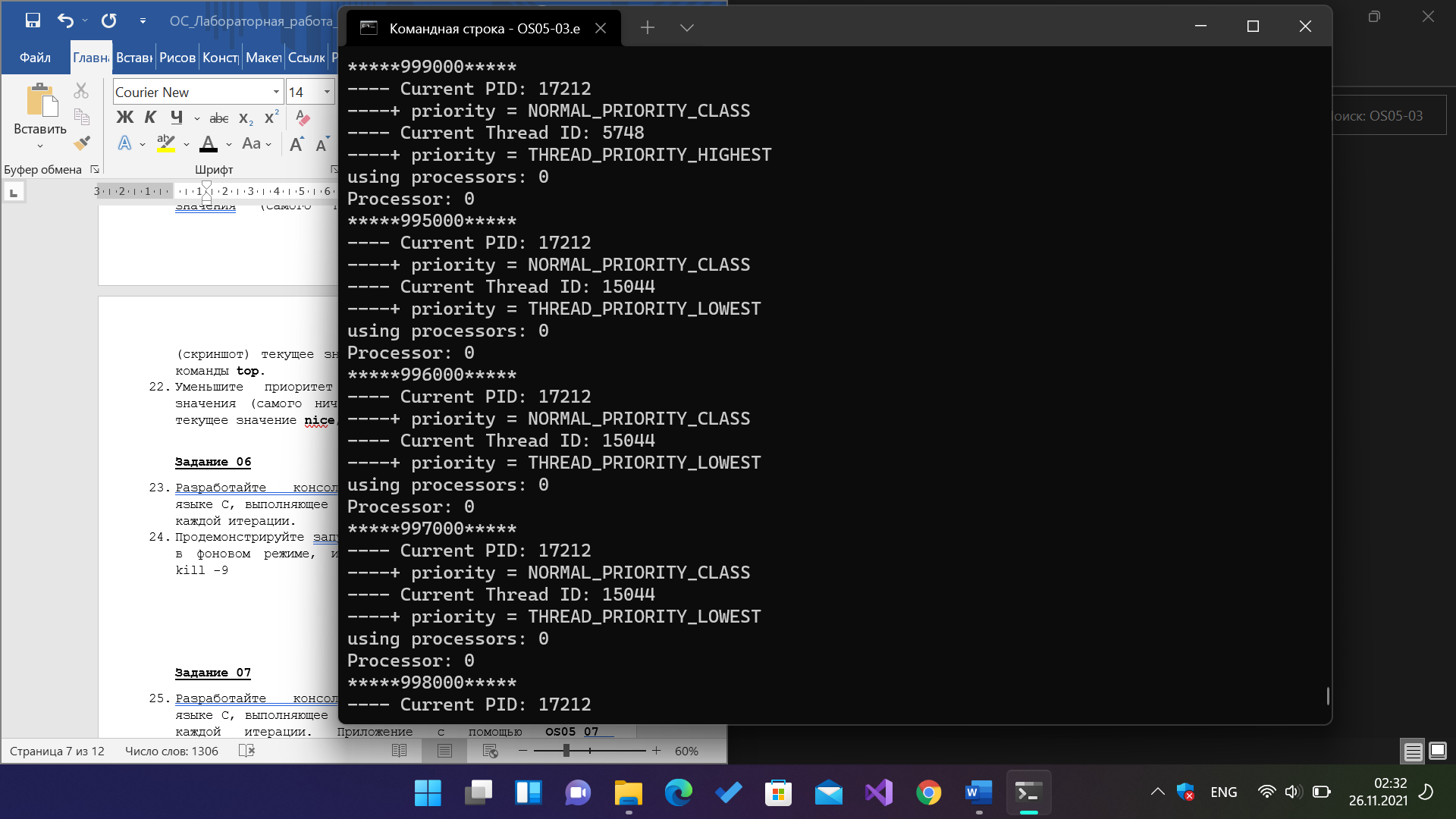


Зафиксируйте (скриншот) в момент первого окончания одного из дочерних потоков расхождение в количестве выполненных потоками итераций.

1. Запустите приложение **OS05\_03**,принимающее следующие значения параметров:

* P1: доступен один процессор;
* P2: Normal;
* P3: Lowest;
* P4: Highest.

(1 32 -2 2)



Зафиксируйте (скриншот) в момент первого окончания одного из дочерних потоков расхождение в количестве выполненных потоками итераций.

1. По зафиксированным скриншотам, объясните полученные результаты.

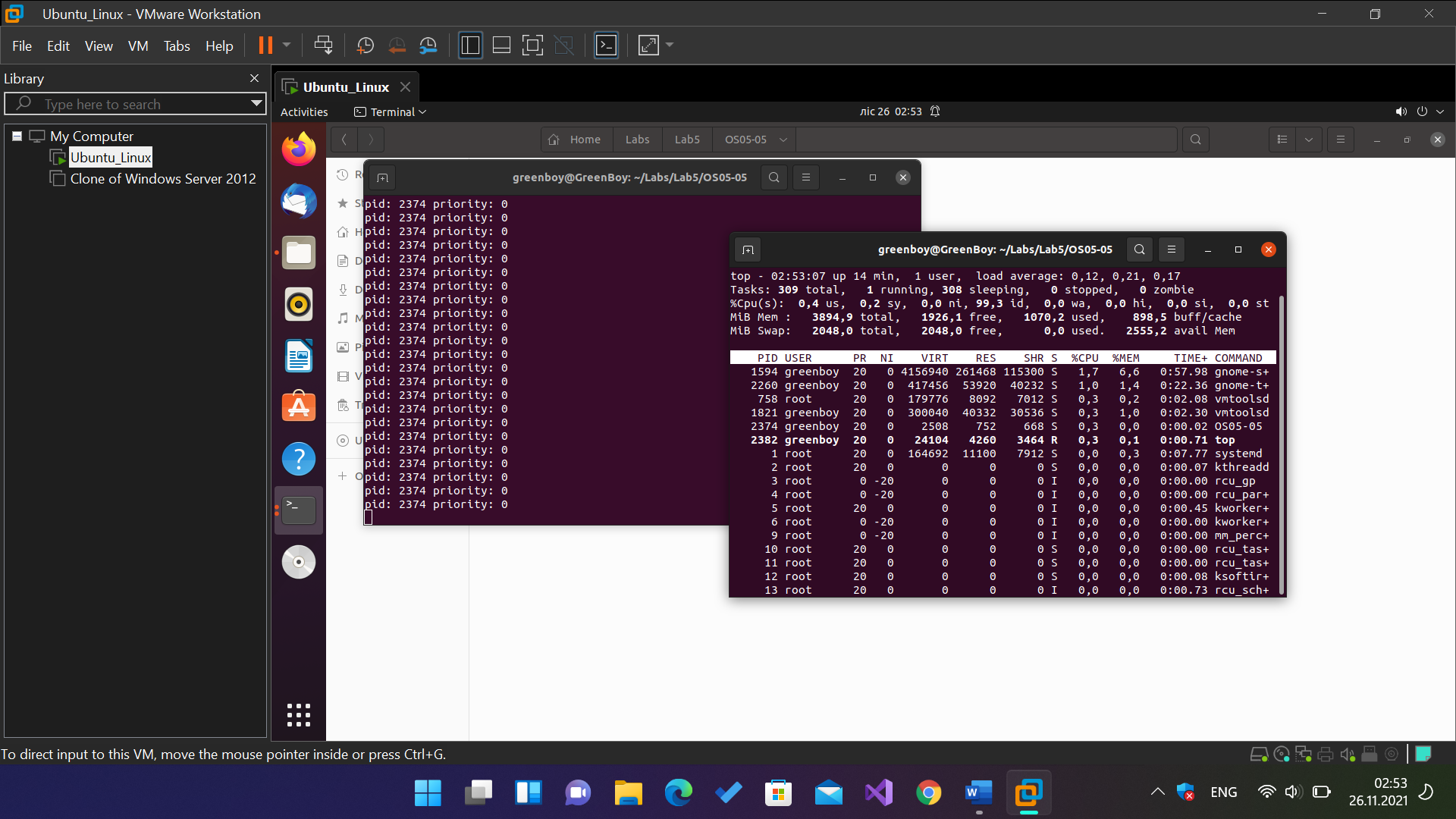
**Задание 04**

1. Разработайте консольное Linux-приложение **OS05\_04** на языке С++, выводящее на консоль следующую информации:

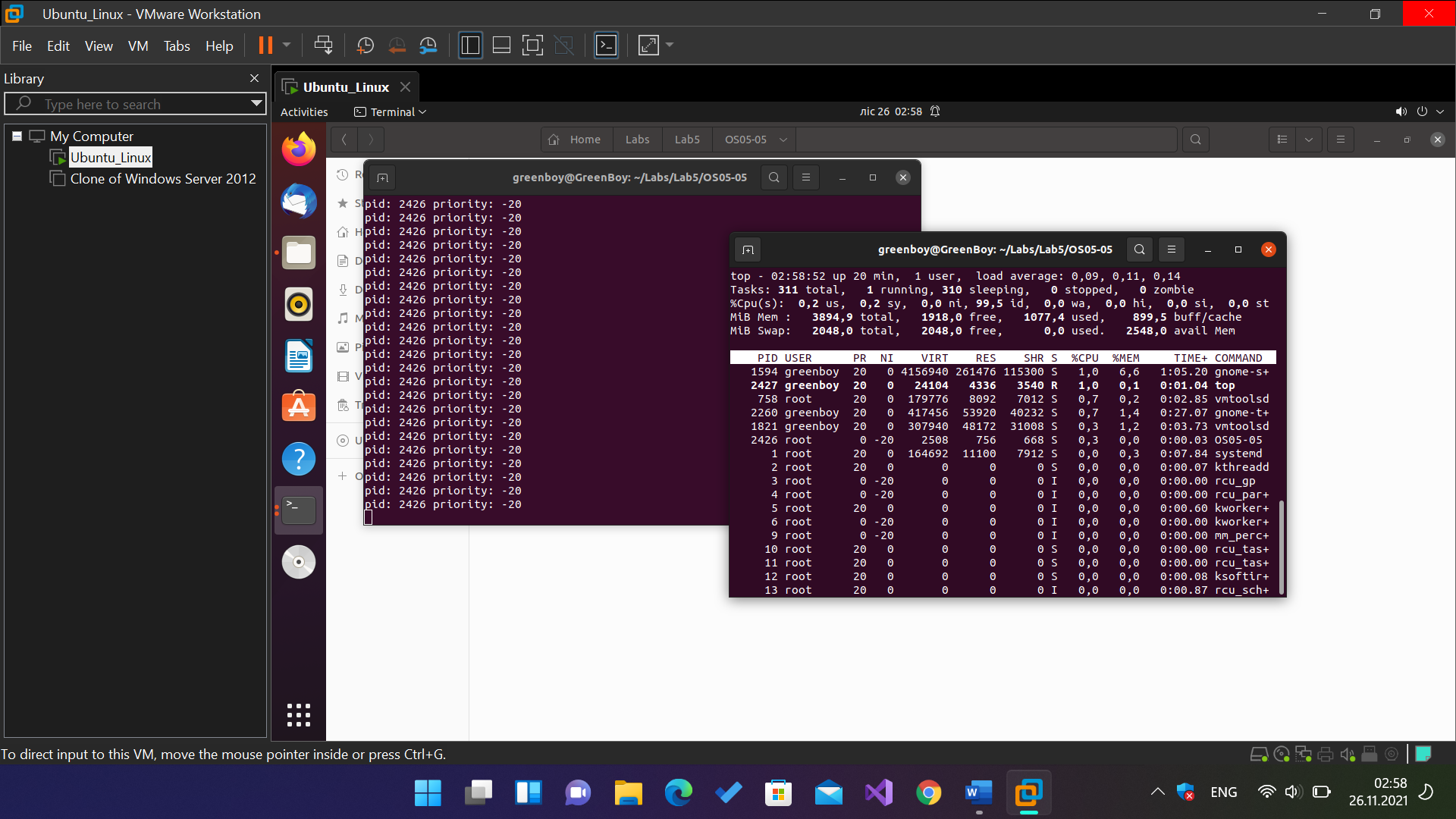
* идентификатор текущего процесса;
* идентификатор текущего (main) потока;
* приоритет (nice) текущего потока;
* номера доступных процессоров.

**Задание 05**

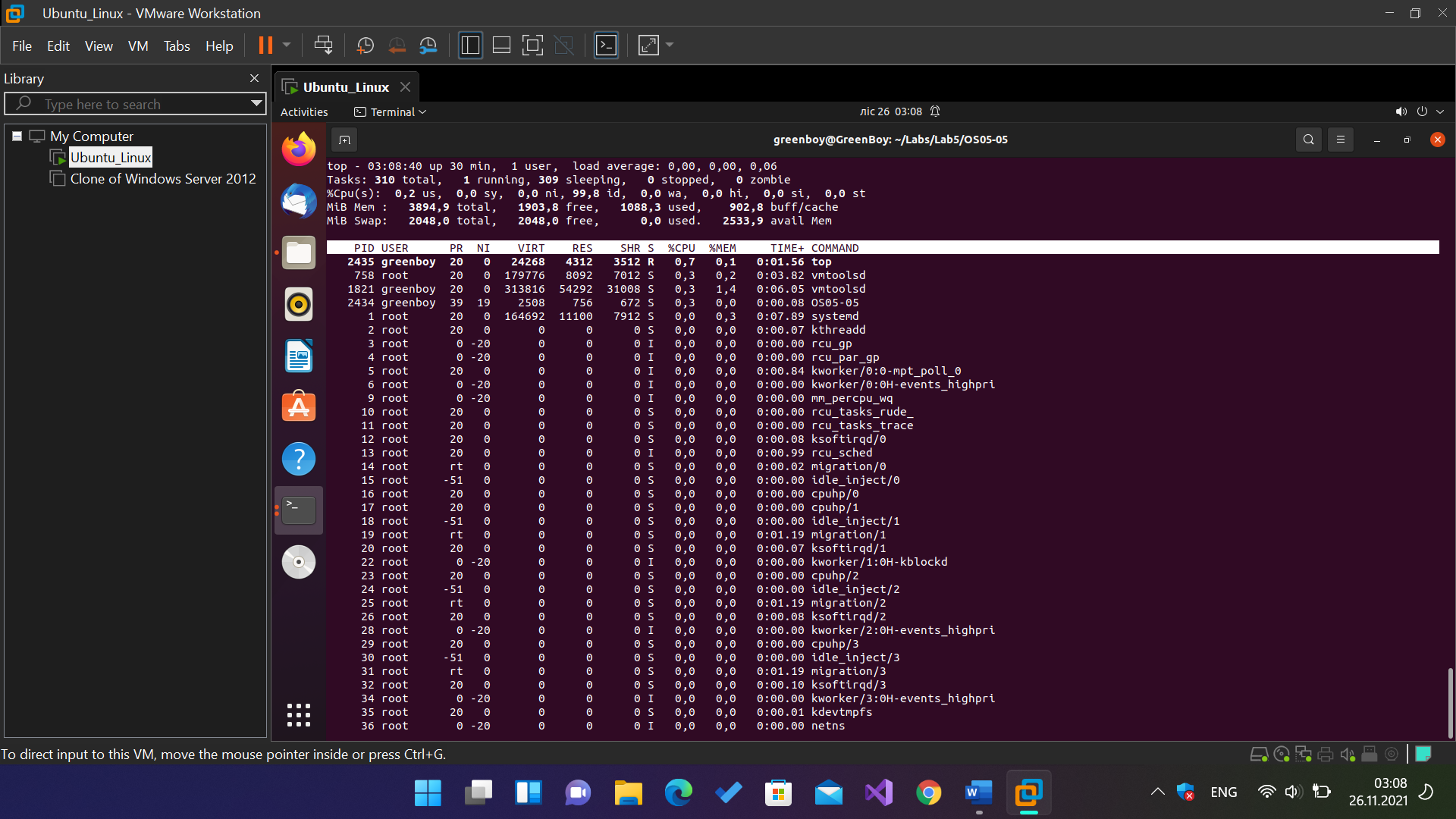
1. Разработайте консольное Linux-приложение **OS05\_05** на языке С, выполняющее длинный цикл.
2. Запустите приложение **OS05\_05.**
3. Зафиксируйте (скриншот) текущее значение **nicе**, полученное с помощью команды **top**.



1. Увеличьте приоритет для **OS05\_05** до максимального значения (самого привилегированного). Зафиксируйте (скриншот) текущее значение **nicе**, полученное с помощью команды **top**.



1. Уменьшите приоритет для **OS05\_05** до минимального значения (самого ничтожного). Зафиксируйте (скриншот) текущее значение **nicе**, полученное с помощью команды **top**



**Задание 06**

1. Разработайте консольное Linux-приложение **OS05\_06** на языке С, выполняющее длинный цикл с задержкой в 1сек в каждой итерации.
2. Продемонстрируйте запуск нескольких приложения **OS05\_06** в фоновом режиме, и команды bg, fg, jobs, Ctrl+Z, kill -9

**Задание 07**

1. Разработайте консольное Linux-приложение **OS05\_07** на языке С, выполняющее длинный цикл с задержкой в 1сек в каждой итерации. Приложение с помощью **OS05\_07** системного вызова **fork** вызывает дочерний поток который понижает свой приоритет на 10.
2. С помощью команды **watch ps** продемонстрируйте работу этих потоков и их значение **nice**.

**Задание 08.**Ответьте на следующие вопросы

1. Поясните понятие «мультизадачная OS с вытеснением».

ОС, c одновременным выполнением нескольких задач, в которой планирование процессов основывается на абсолютных приоритетах. Процесс с меньшим приоритетом может быть вытеснен при его выполнении более приоритетным процессом

1. Поясните понятие «циклическое планирование»

Каждому процессу предоставляется квант времени процессора. Когда квант заканчивается процесс переводится планировщиком в конец очереди. При блокировке процесс выпадает из очереди.

1. Поясните понятие «приоритетное планирование».

Каждому процессу присваивается **приоритет**, и процессу выделяется процессорное время в зависимости от приоритета.

1. Поясните понятие «кооперативное планирование».

Управление не передается другому процессу пока текущий процесс не будет готов отдать управление другому процессу

1. Поясните понятие «OS реального времени».

Тип ОС которая гарантирует своевременное выполнение какой-либо задачи за ограниченный промежуток времени

1. Поясните понятие «приоритет процесса».

Значение, определяющее как часто данный процесс по сравнению с другими процессами будет исполняться процессором

1. Поясните выражение «поток уступает процессор другому потоку».

Процессор переключается на следующий поток, готовый к выполнению

1. Windows: как поток может уступить процессор?

С помощью системного вызова sleep(0)

1. Windows: что такое базовый приоритет потока, как он вычисляется и диапазон его изменения?

Базовый приоритет — это то **значение**, чем будет являться приоритет потока с типом приоритета **Normal**:

Вычисляется по таблице на основе приоритета процесса

От 0 до 31

1. Windows: поясните назначение и принцип применения системного вызова SetThreadIdealProcessor.

устанавливает предпочитаемый процессор для потока

первый параметр дескриптор потока

второй номер процессора

Значение **MAXIMUM\_PROCESSORS** сообщает системе, что поток не имеет привилегированного процессора.

1. Windows: поясните назначение и принцип применения системного вызова ResumeThread.

Служит для продолжения выполнения потока

1. Windows: поясните назначение и принцип применения системного вызова WaitForSingleObject.

Ожидает, пока указанный объект не перейдет в сигнальное состояние или не истечет интервал ожидания.

1. Windows: поясните назначение и принцип применения системных вызовов GetProcessPriorityBoost, GetThreadPriorityBoost, SetProcessPriorityBoost, SetThreadPriorityBoost.

GetProcessPriorityBoost – Извлекает состояние управления повышением приоритета указанного процесса.

GetThreadPriorityBoost - извлекает состояние управления повышением приоритета указанного потока.

SetProcessPriorityBoost - отключает или включает возможность системы временно повышать приоритет потоков указанного процесса.

SetThreadPriorityBoost - Отключает или включает возможность системы временно повышать приоритет потока.

1. Linux: поясните принцип идентификации процессов и потоков и поясните почему он такой.

Идентификатор главного потока равен идентификатору процесса. ID других потоков увеличивается на 1, по сравнению с предыдущим.

1. Linux: Поясните понятие «планировщик потоков».

Это алгоритм, который определяет порядок выполнения потоков.

Планировщик — это часть ядра, которая решает какая запущенная нить будет выполняться процессором следующей.

1. Linux: поясните принцип использования значения nice –процесса, диапазон его изменения, для какого режима работы планировщика это значение применяется?

Nice определяет приоритетность при политике OTHER. Диапазон от -20 до 19. Чем меньше nice, тем лучше

1. Linux: перечислите политики планирования, какая действует по умолчанию?

1 стандартная (OTHER)

2 FIFO – политика

3 карусельная политика(RR)

4 пакетная политика(BATCH)

1. Linux: как выяснить действующую политику планирования для процесса с помощью файловой системы proc?

Proc/%pid%/shed

0- OTHER

1-FIFO

2-Карусельная политика (RR)

3- пакетная политика (BATCH)

1. Linux: с помощью какого системного вызова поток может уступить процессор.

**shed\_yield()**

1. Linux: чем отличается системный вызов **nice** от вызова **setpriority**.

Nice изменяет значение nice текущего процесса

1. Linux: поясните понятие «планировщик ввода вывода», каким образом можно выяснить какие планировщики ввода/ вывода доступны?

программная прослойка между блочными устройствами (дисковые устройства) и низкоуровневыми драйверами ввода/вывода.

Dmesq | grep scheduler

1. Linux: перечислите известные вам планировщики ввода/ вывода, кратко охарактеризуйте их.

Noop – простой планировщик, объединяет однотипные запросы для сокращения операций

CFQ – у каждого процесса своя очередь, у каждой очереди свой квант времени

Deadline – пытается выполнить запрос в указанное время, две очереди read & write. Read приоритетнее

BFQ – каждой cfq очереди выделяется бюджет, который растет для процессов с интенсивным вводом/выводом

MQ-Deadline – модификация deadline для новых устройств

kyber – для работы с быстрыми устройствами, две очереди read & write , read приоритетнее

1. Linux: каким образом можно выяснить тип планировщика действующего для блокового устройства?

cat /sys/block/sda/queue/scheduler