**Задание 01**

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

- идентификатор текущего процесса;

- идентификатор текущего (main) потока;

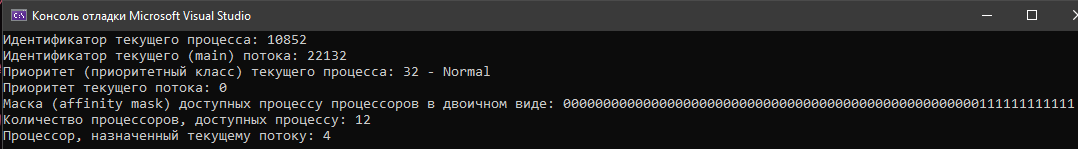
- приоритет (приоритетный класс) текущего процесса;

- приоритет текущего потока;

- маску (affinity mask) доступных процессу процессоров в двоичном виде;

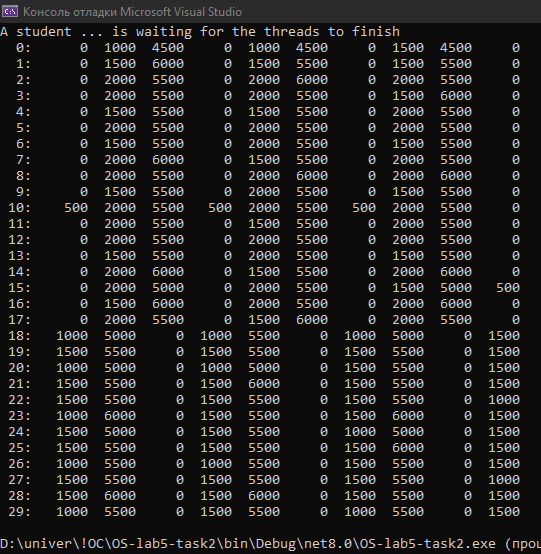
- количество процессоров, доступных процессу;

- процессор, назначенный текущему потоку.



**Задание 02**

2. Создайте консольное Windows OS05\_02 на языке С#, 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).



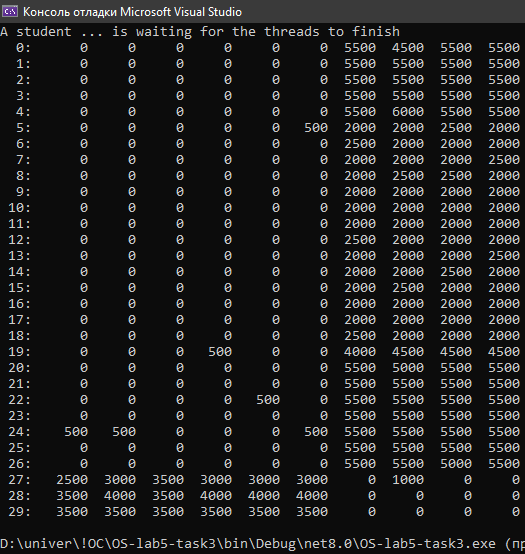
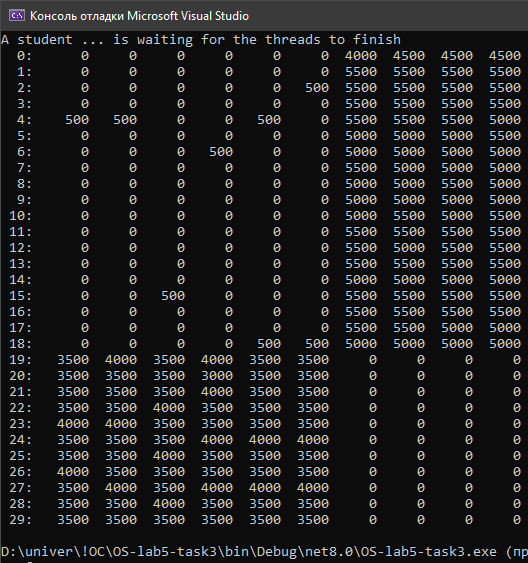
**Задание 03**

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

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**

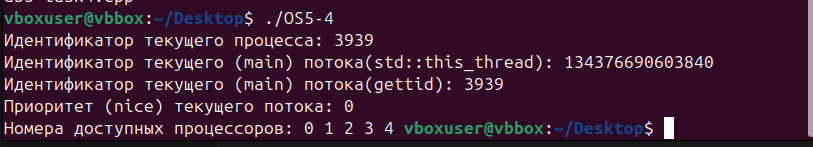
9. Разработайте консольное Linux-приложение OS05\_04 на языке С++, выводящее на консоль следующую информации:

- идентификатор текущего процесса;

- идентификатор текущего (main) потока;

- приоритет (nice) текущего потока;

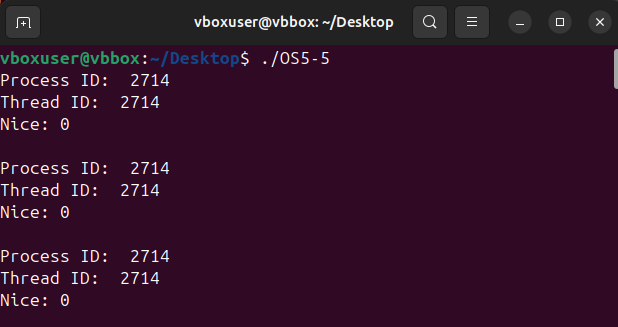
- номера доступных процессоров.



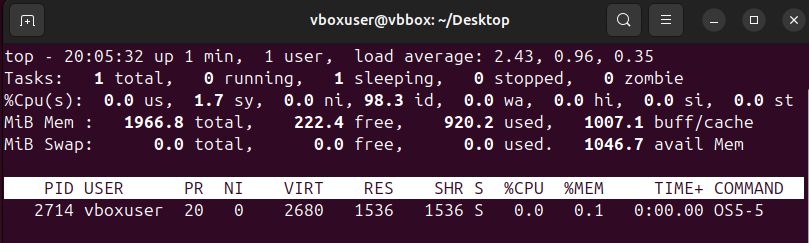
**Задание 05**

10. Разработайте консольное Linux-приложение OS05\_05 на языке С, выполняющее длинный цикл.

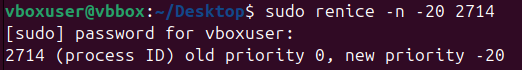
11. Запустите приложение OS05\_05.

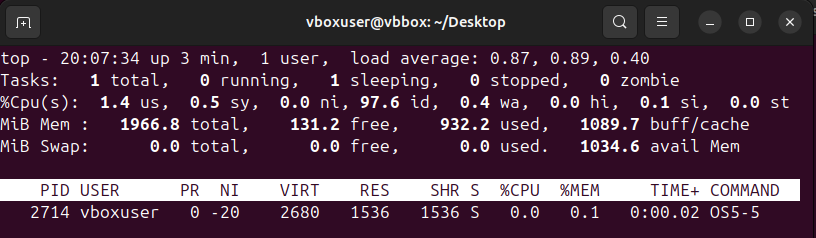


12. Зафиксируйте < > текущее значение nicе, полученное с помощью команды top.

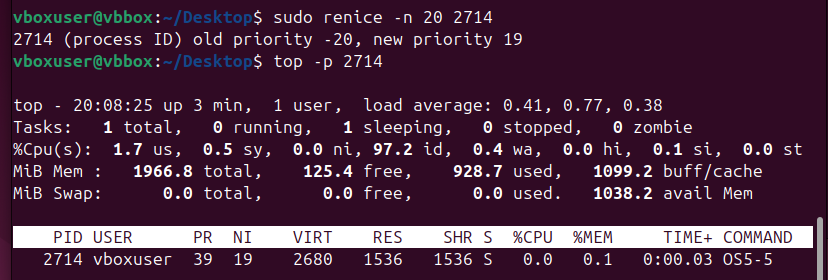


13. Увеличьте приоритет для OS05\_05 до максимального значения (самого привилегированного). Зафиксируйте < > текущее значение nicе, полученное с помощью команды top.





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



**Ответьте на следующие вопросы**

1. Поясните понятие «мультизадачная OS с вытеснением».

Вид многозадачности, в котором операционная система сама передает управление от одной выполняемой программы другой в случае завершения операций ввода-вывода, возникновения событий в аппаратуре компьютера, истечения таймеров и квантов времени, или же поступлений тех или иных сигналов от одной программы к другой.

1. Поясните понятие «циклическое планирование».

Каждому процессу назначается определенный интервал времени, называемый его квантом, в течение которого ему предоставляется возможность выполнения. Если процесс к завершению кванта времени все еще выполняется, то ресурс центрального процессора у него отбирается и передается другому процессу.

1. Поясните понятие «приоритетное планирование».  
   каждому процессу присваивается значение приоритетности и запускается тот процесс, который находится в состоянии готовности и имеет наивысший приоритет.
2. Поясните понятие «кооперативное планирование».

Тип многозадачности, при котором следующая задача выполняется только после того, как текущая задача явно объявит себя готовой отдать процессорное время другим задачам.

1. Поясните понятие «OS реального времени».

это способность операционной системы обеспечить требуемый уровень сервиса в определённый промежуток времени.

1. Поясните понятие «приоритет процесса».

Приоритет процесса означает, насколько больше процессорного времени будет отдано этому процессу по сравнению с другими

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

Один поток завершает своё выполнение или временно приостанавливает выполнение, позволяя другому потоку получить доступ к процессору.

1. Windows: как поток может уступить процессор?

OS: Windows, уступить процессор Sleep(0)

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

Класс приоритета процесса и уровень приоритета потока объединяются для формирования базового приоритета каждого потока.

Базовый приоритет потока зависит от приоритета родительского процесса. При создании нового потока его базовый приоритет устанавливается равным приоритету процесса.

В Windows базовый приоритет потока может варьироваться от 1 (низший приоритет) до 31 (высший приоритет).

Real-time: от 16 до 31

High: от 13 до 15

Above Normal: от 10 до 12

Normal: от 8 до 9

Below Normal: от 7 до 7

Idle: 4

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

Возобновляет выполнения потока, который был приостановлен

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

Ожидает завершения работы одного объекта синхронизации, такого как поток, событие, мьютекс или семафор.

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

GetProcessPriorityBoost, GetThreadPriorityBoost - функции используются для получения текущего значения приоритетного повышения для процесса или потока соответственно.

SetProcessPriorityBoost, SetThreadPriorityBoost - функции используются для включения или выключения приоритетного повышения для процесса или потока соответственно.

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

Каждый процесс в Linux имеет уникальный идентификатор, называемый PID (Process ID). Этот идентификатор используется для управления процессами и взаимодействия с ними.

В Linux потоки рассматриваются как отдельные процессы с общим адресным пространством. Каждый поток также имеет уникальный идентификатор, называемый TID (Thread ID).

TID в Linux является тем же самым, что и PID, поскольку Linux использует модель "потоков как процессов".

1. Linux: Поясните понятие «планировщик потоков».

OS: process scheduler – планировщик процессов в Linux, компонент ядра, выбирает процесс для выполнения, квантует процессорное время, ˜20мс., O(1)-планировщик(shed.c)

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

nice - это значение "приятности", которое варьируется от -20 (самый высокий приоритет) до 19 (самый низкий приоритет)

Приоритет nice и приоритет планировщика процессов ядра ОС — разные числа. Число nice — приоритет, который пользователь хотел бы назначить процессу. Приоритет планировщика — действительный приоритет, назначенный процессу планировщиком. Обычно, NI = PRI - 20, но это верно не всегда. По умолчанию NI=0, соответственно PRI=20.

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

OS: Linux, политики планирования процесса: 1) стандартная (OTHER, разделения времени); 2) FIFO-политика (FIFO, реального времени); 3) карусельная (round-robin) политика (RR); 4) пакетная политика(BATCH).

1. Linux: с помощью какого системного вызова поток может уступить процессор. sched\_yield()

**Попроще:**

1. Мультизадачная ОС с вытеснением

Это система, которая позволяет запускать несколько программ одновременно. Операционная система сама решает, когда переключить выполнение с одной программы на другую. Это происходит, если одна программа ждет ввода-вывода или если истекло отведенное ей время.

2. Циклическое планирование

Каждой программе дается определенное время (квант), чтобы она могла работать. Если программа не завершает свою работу за это время, управление передается следующей программе. Это помогает всем программам получать время для выполнения.

3. Приоритетное планирование

Каждой программе присваивается приоритет. Система запускает ту программу, у которой приоритет выше. Это помогает важным задачам выполняться быстрее.

4. Кооперативное планирование

В этой системе программа сама решает, когда уступить управление другим задачам. То есть, одна программа должна завершить или приостановить свою работу, прежде чем другая начнет выполняться.

5. ОС реального времени

Это операционная система, которая должна выполнять задачи в строго определенные сроки. Например, если требуется, чтобы система реагировала на события в течение 1 миллисекунды, она должна это гарантировать.

6. Приоритет процесса

Приоритет процесса определяет, сколько времени процессор будет выделять этому процессу по сравнению с другими. Процессы с высоким приоритетом получают больше времени на выполнение.

7. Поток уступает процессор другому потоку

Когда один поток завершает свою работу или временно останавливается, он позволяет другому потоку использовать процессор. Это происходит для того, чтобы все потоки могли работать.

8. Как поток может уступить процессор в Windows?

В Windows поток может временно «успокоиться» (вызвав Sleep(0)), что позволяет другим потокам получить доступ к процессору.

9. Базовый приоритет потока в Windows

Базовый приоритет потока определяется приоритетом процесса, к которому он принадлежит. Он может варьироваться от 1 (низкий приоритет) до 31 (высокий приоритет). Например:

Реальное время: 16-31

Высокий: 13-15

Обычный: 8-9

10. Системный вызов ResumeThread в Windows

Этот вызов используется для возобновления работы потока, который был приостановлен. Он активирует поток, чтобы он продолжил выполнение.

11. Системный вызов WaitForSingleObject в Windows

Этот вызов заставляет программу ждать завершения работы определенного объекта, например, потока или события. Это полезно для синхронизации.

12. Системные вызовы Get/SetProcessPriorityBoost и Get/SetThreadPriorityBoost

GetProcessPriorityBoost и GetThreadPriorityBoost используются для получения текущего значения приоритетного повышения.

SetProcessPriorityBoost и SetThreadPriorityBoost устанавливают, будет ли приоритет повышен или нет.

13. Идентификация процессов и потоков в Linux

В Linux каждый процесс имеет уникальный идентификатор (PID), который используется для управления им. Потоки рассматриваются как отдельные процессы с общим адресным пространством и имеют свои идентификаторы (TID).

14. Планировщик потоков в Linux

Это компонент ядра, который выбирает, какой процесс будет выполняться в данный момент. Он распределяет процессорное время между процессами, обычно около 20 мс на каждую задачу.

15. Значение nice в Linux

nice — это значение, которое показывает желаемый приоритет процесса. Оно варьируется от -20 (высокий приоритет) до 19 (низкий приоритет). Приоритет процесса не всегда совпадает с его значением nice.

16. Политики планирования в Linux

В Linux есть несколько политик планирования:

Стандартная (обычная)

FIFO (реальное время)

Карусельная (round-robin)

Пакетная (batch)

По умолчанию используется стандартная политика.

17. Как поток может уступить процессор в Linux

В Linux поток может вызвать sched\_yield(), чтобы уступить процессор другим потокам.