Лабораторная работа 05

ОС, ПОИТ-3

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. Разработайте консольное Windows-приложение **OS05_02x**, выполняющее цикл в 1млн итераций.
3. 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.
4. 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.
5. 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.

6. Запустите приложение **OS05_02,** принимающее следующие значения параметров:
   - P1: доступны все процессоры;
   - P2: Normal;
   - P3: Normal.

   Зафиксируйте (скриншот) в момент первого окончания одного из дочерних процессов расхождение в количестве выполненных процессами итераций.
7. Запустите приложение **OS05_02,** принимающее следующие значения параметров:
   - P1: доступны все процессоры;
   - P2: Below Normal;
   - P3: High.

   Зафиксируйте (скриншот) в момент первого окончания одного из дочерних процессов расхождение в количестве выполненных процессами итераций.
8. Запустите приложение **OS05_02,** принимающее следующие значения параметров:
   - P1: доступен один процессор;
   - P2: Below Normal;
   - P3: High.

   Зафиксируйте (скриншот) в момент первого окончания одного из дочерних процессов расхождение в количестве выполненных процессами итераций.
9. По зафиксированным скриншотам, объясните полученные результаты.

Задание 03

10. 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.

11. Приложение **OS05_03** включает в себя потоковую функцию **TA**, выполняющую цикл в 1млн итераций, аналогичный циклу в задании 02.
12. 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.
13. Запустите приложение **OS05_03**, принимающее следующие значения параметров:
    - P1: доступны все процессоры;
    - P2: Normal;
    - P3: Normal;
    - P4: Normal;

    Зафиксируйте (скриншот) в момент первого окончания одного из дочерних потоков расхождение в количестве выполненных потоками итераций.
14. Запустите приложение **OS05_03**, принимающее следующие значения параметров:
    - P1: доступны все процессоры;
    - P2: Normal;
    - P3: Lowest;
    - P4: Highest.

    Зафиксируйте (скриншот) в момент первого окончания одного из дочерних потоков расхождение в количестве выполненных потоками итераций.
15. Запустите приложение **OS05_03**, принимающее следующие значения параметров:
    - P1: доступен один процессор;
    - P2: Normal;
    - P3: Lowest;
    - P4: Highest.

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

16. По зафиксированным скриншотам объясните полученные результаты.

Задание 04

17. 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

18. Разработайте консольное Linux-приложение **OS05_05** на языке C, выполняющее длинный цикл.
19. Запустите приложение **OS05_05.**
20. Зафиксируйте (скриншот) текущее значение **nice**, полученное с помощью команды **top.**
21. Увеличьте приоритет для **OS05_05** до максимального значения (самого привилегированного). Зафиксируйте (скриншот) текущее значение **nice**, полученное с помощью команды **top.**
22. Уменьшите приоритет для **OS05_05** до минимального значения (самого ничтожного). Зафиксируйте (скриншот) текущее значение **nice**, полученное с помощью команды **top**

Задание 06

23. Разработайте консольное Linux-приложение **OS05_06** на языке C, выполняющее длинный цикл с задержкой в 1сек в каждой итерации.
24. Продемонстрируйте запуск нескольких приложения **OS05_06** в фоновом режиме, и команды bg, fg, jobs, Ctrl+Z, kill -9

Задание 07

25. Разработайте консольное Linux-приложение **OS05_07** на языке C, 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.
26. С помощью команды **watch ps** продемонстрируйте работу этих потоков и их значение **nice**.

Задание 08.Ответьте на следующие вопросы

27. Поясните понятие «мультизадачная OS с вытеснением».
28. Поясните понятие «циклическое планирование».
29. Поясните понятие «приоритетное планирование».
30. Поясните понятие «кооперативное планирование».
31. Поясните понятие «OS реального времени».
32. Поясните понятие «приоритет процесса».
33. Поясните выражение «поток уступает процессор другому потоку».
34. Windows: как поток может уступить процессор?
35. Windows: что такое базовый приоритет потока, как он вычисляется и диапазон его изменения?
36. Windows: поясните назначение и принцип применения системного вызова SetThreadIdealProcessor.
37. Windows: поясните назначение и принцип применения системного вызова ResumeThread.
38. Windows: поясните назначение и принцип применения системного вызова WaitForSingleObject.
39. Windows: поясните назначение и принцип применения системных вызовов GetProcessPriorityBoost, GetThreadPriorityBoost, SetProcessPriorityBoost, SetThreadPriorityBoost.
40. Linux: поясните принцип идентификации процессов и потоков и поясните почему он такой.
41. Linux: Поясните понятие «планировщик потоков».
42. Linux: поясните принцип использования значения nice – процесса, диапазон его изменения, для какого режима работы планировщика это значение применяется?
43. Linux: перечислите политики планирования, какая действует по умолчанию?

44. Linux: как выяснить действующую политику планирования для процесса с помощью файловой системы proc?
45. Linux: с помощью какого системного вызова поток может уступить процессор.
46. Linux: чем отличается системный вызов **nice** от вызова **setpriority**.
47. Linux: поясните понятие «планировщик ввода вывода», каким образом можно выяснить какие планировщики ввода/вывода доступны?
48. Linux: перечислите известные вам планировщики ввода/вывода, кратко охарактеризуйте их.
49. Linux: каким образом можно выяснить тип планировщика действующего для блокового устройства?
50.