МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Операционные системы»

Исполнитель

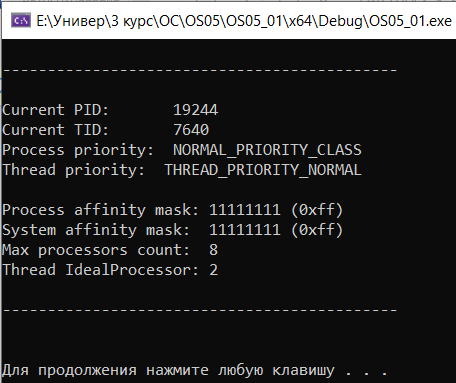
студент (ка) 3 курса группы 6 Розель Станислав Александрович

(Ф.И.О.)

Минск 2024

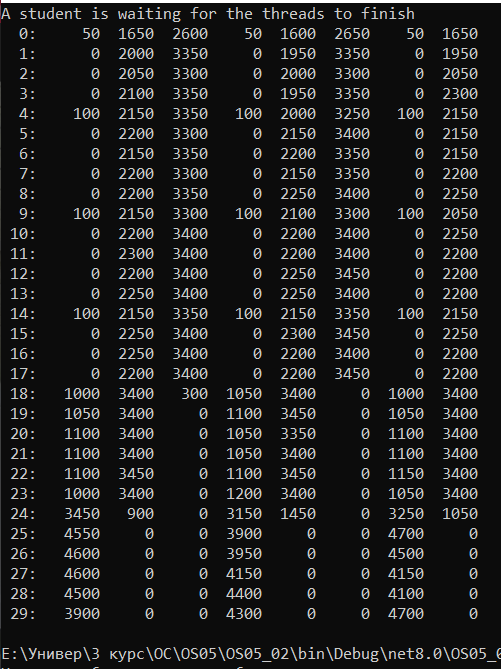
**Задание 01**

|  |
| --- |
| #include <iostream>  #include <cstdlib>  #include <intrin.h>  #include <iomanip>  #include "Windows.h"  #include <stdexcept>  using namespace std;  void PrintProcessPriority(HANDLE processHandle)  {  try  {  DWORD processPriority = GetPriorityClass(processHandle);  switch (processPriority)  {  case IDLE\_PRIORITY\_CLASS: cout << "Process priority: IDLE\_PRIORITY\_CLASS\n"; break;  case BELOW\_NORMAL\_PRIORITY\_CLASS: cout << "Process priority: BELOW\_NORMAL\_PRIORITY\_CLASS\n"; break;  case NORMAL\_PRIORITY\_CLASS: cout << "Process priority: NORMAL\_PRIORITY\_CLASS\n"; break;  case ABOVE\_NORMAL\_PRIORITY\_CLASS: cout << "Process priority: ABOVE\_NORMAL\_PRIORITY\_CLASS\n"; break;  case HIGH\_PRIORITY\_CLASS: cout << "Process priority: HIGH\_PRIORITY\_CLASS\n"; break;  case REALTIME\_PRIORITY\_CLASS: cout << "Process priority: REALTIME\_PRIORITY\_CLASS\n"; break;  default: throw std::runtime\_error("[ERROR] Unknown process priority.");  }  }  catch (const std::exception& ex)  {  cout << ex.what() << endl;  }  }  void PrintThreadPriority(HANDLE threadHandle)  {  try  {  DWORD threadPriority = GetThreadPriority(threadHandle);  switch (threadPriority)  {  case THREAD\_PRIORITY\_LOWEST: cout << "Thread priority: THREAD\_PRIORITY\_LOWEST\n"; break;  case THREAD\_PRIORITY\_BELOW\_NORMAL: cout << "Thread priority: THREAD\_PRIORITY\_BELOW\_NORMAL\n"; break;  case THREAD\_PRIORITY\_NORMAL: cout << "Thread priority: THREAD\_PRIORITY\_NORMAL\n"; break;  case THREAD\_PRIORITY\_ABOVE\_NORMAL: cout << "Thread priority: THREAD\_PRIORITY\_ABOVE\_NORMAL\n"; break;  case THREAD\_PRIORITY\_HIGHEST: cout << "Thread priority: THREAD\_PRIORITY\_HIGHEST\n"; break;  case THREAD\_PRIORITY\_IDLE: cout << "Thread priority: THREAD\_PRIORITY\_IDLE\n"; break;  default: throw std::runtime\_error("[ERROR] Unknown thread priority.");  }  }  catch (const std::exception& ex)  {  cout << ex.what() << endl;  }  }  void PrintAffinityMask(HANDLE hp, HANDLE ht)  {  try  {  DWORD\_PTR pa = NULL, sa = NULL, icpu = -1;  char buf[32];  if (!GetProcessAffinityMask(hp, &pa, &sa))  throw std::runtime\_error("[FATAL] GetProcessAffinityMask threw an exception.");  \_itoa\_s(pa, buf, 2);  cout << "Process affinity mask: " << buf;  cout << " (" << showbase << hex << pa << ")\n";  \_itoa\_s(sa, buf, 2);  cout << "System affinity mask: " << buf;  cout << " (" << showbase << hex << sa << ")\n";  SYSTEM\_INFO sys\_info;  GetSystemInfo(&sys\_info);  cout << "Max processors count: " << dec << sys\_info.dwNumberOfProcessors << "\n";  icpu = SetThreadIdealProcessor(ht, MAXIMUM\_PROCESSORS);  cout << "Thread IdealProcessor: " << dec << icpu << "\n";  }  catch (const std::exception& ex)  {  cout << ex.what() << endl;  }  }  int main()  {  HANDLE processHandle = GetCurrentProcess();  HANDLE threadHandle = GetCurrentThread();  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  cout << "\n--------------------------------------------\n\n";  cout << "Current PID: " << pid << "\n";  cout << "Current TID: " << tid << "\n";  PrintProcessPriority(processHandle);  PrintThreadPriority(threadHandle);  cout << "\n";  PrintAffinityMask(processHandle, threadHandle);  cout << "\n--------------------------------------------\n\n\n";  system("pause");  } |



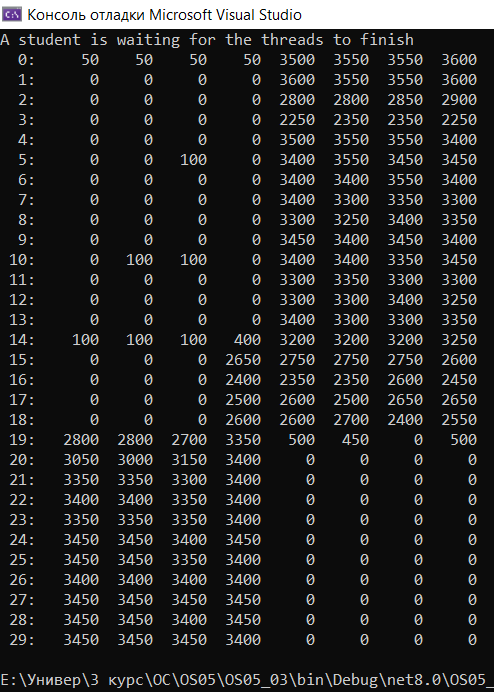
**Задание 02**

|  |
| --- |
| using System;  using System.Diagnostics;  using System.Diagnostics.CodeAnalysis;  using System.Runtime.CompilerServices;  class Program  {  const int ThreadCount = 8;  const int ThreadLifeTime = 60;  const int ObservationTime = 30;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime = DateTime.Now;  static void WorkThread(object o)  {  int id = (int)o;  for (int i = 0; i < ThreadLifeTime \* 20; i++)  {  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)Math.Round(CurrentTime.Subtract(StartTime).TotalSeconds - 0.49);  if (ElapsedSeconds >= 0 && ElapsedSeconds < ObservationTime)  {  Matrix[id, ElapsedSeconds] += 50;  }  MySleep(50); // Из задания 5  }  }  static void Main(string[] args)  {  Process.GetCurrentProcess().ProcessorAffinity = (System.IntPtr)15;  Thread[] t = new Thread[ThreadCount];  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);  }  Console.WriteLine("A student is waiting for the threads to finish");  for (int i = 0; i < ThreadCount; ++i)  t[i].Join();  for (int s = 0; s < ObservationTime; s++)  {  Console.Write("{0,3}: ", s);  for (int th = 0; th < ThreadCount; th++)  {  Console.Write(" {0,5}", Matrix[th, s]);  }  Console.WriteLine();  }  }  static Double MySleep(int ms)  {  Double Sum = 0, Temp;  for (int t = 0; t < ms; ++t)  {  Temp = 0.711 + (Double)t / 10000.0;  Double a, b, c, d, e, nt;  for (int k = 0; k < 5500; ++k)  {  nt = Temp - k / 27000.0;  a = Math.Sin(nt);  b = Math.Cos(nt);  c = Math.Cos(nt / 2.0);  d = Math.Sin(nt / 2);  e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);  Sum += e;  }  }  return Sum;  }  } |



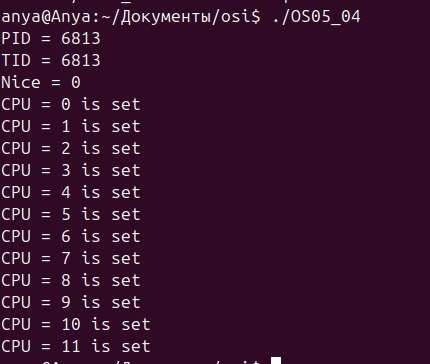
**Задание 03**

|  |
| --- |
| using System;  using System.Diagnostics;  using System.Diagnostics.CodeAnalysis;  using System.Runtime.CompilerServices;  class Program  {  const int ThreadCount = 8;  const int ThreadLifeTime = 60;  const int ObservationTime = 30;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime = DateTime.Now;  static void WorkThread(object o)  {  int id = (int)o;  for (int i = 0; i < ThreadLifeTime \* 20; i++)  {  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)Math.Round(CurrentTime.Subtract(StartTime).TotalSeconds - 0.49);  if (ElapsedSeconds >= 0 && ElapsedSeconds < ObservationTime)  {  Matrix[id, ElapsedSeconds] += 50;  }  MySleep(50); // Из задания 5  }  }  static void Main(string[] args)  {  Process.GetCurrentProcess().ProcessorAffinity = (System.IntPtr)15;  Thread[] t = new Thread[ThreadCount];  for (int i = 0; i < ThreadCount; ++i)  {  object o = i;  t[i] = new Thread(WorkThread);  if (i < 3) // здесь 2 - половина логических процессоров  t[i].Priority = ThreadPriority.Lowest;  else  t[i].Priority = ThreadPriority.Highest;  t[i].Start(o);  }  Console.WriteLine("A student is waiting for the threads to finish");  for (int i = 0; i < ThreadCount; ++i)  t[i].Join();  for (int s = 0; s < ObservationTime; s++)  {  Console.Write("{0,3}: ", s);  for (int th = 0; th < ThreadCount; th++)  {  Console.Write(" {0,5}", Matrix[th, s]);  }  Console.WriteLine();  }  }  static Double MySleep(int ms)  {  Double Sum = 0, Temp;  for (int t = 0; t < ms; ++t)  {  Temp = 0.711 + (Double)t / 10000.0;  Double a, b, c, d, e, nt;  for (int k = 0; k < 5500; ++k)  {  nt = Temp - k / 27000.0;  a = Math.Sin(nt);  b = Math.Cos(nt);  c = Math.Cos(nt / 2.0);  d = Math.Sin(nt / 2);  e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);  Sum += e;  }  }  return Sum;  }  } |



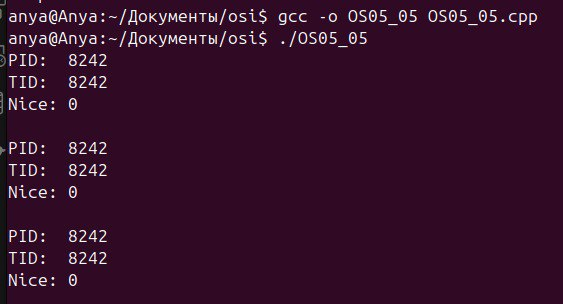
**Задание 04**

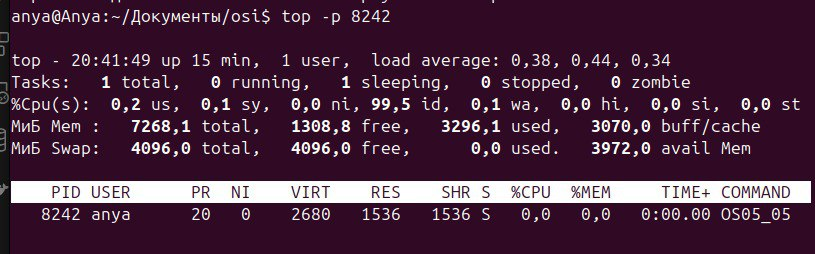
|  |
| --- |
| #include <iostream>  #include <unistd.h>  #include <sys/types.h>  #include <sys/syscall.h>  #include <sys/resource.h>  #include <sched.h>  int main() {  using namespace std;  std::cout << "OS05\_04 Console Application" << std::endl;  pid\_t pid = getpid();  std::cout << "Current process ID: " << pid << std::endl;  std::cout << "Current thread ID: " << syscall(SYS\_gettid) << std::endl;  int currentNiceValue = getpriority(PRIO\_PROCESS, pid);  std::cout << "Current thread priority (nice value): " << currentNiceValue << std::endl;  cpu\_set\_t cpuset;  int cpuCount = sysconf(\_SC\_NPROCESSORS\_ONLN);  CPU\_ZERO(&cpuset);  sched\_getaffinity(0, sizeof(cpuset), &cpuset);  std::cout << "Available CPUs: ";  for (int i = 0; i < cpuCount; i++) {  if (CPU\_ISSET(i, &cpuset)) {  std::cout << i << " ";  }  }  std::cout << std::endl;  return 0;  } |



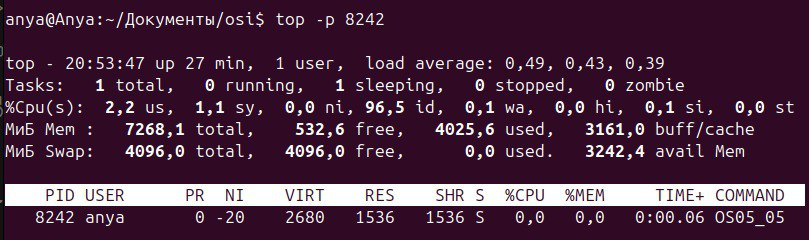
**Задание 05**

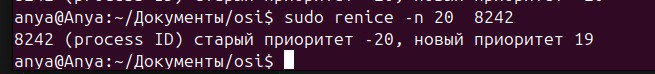
|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  #include <sys/resource.h>  #include <iostream>  int main() {  std::cout<<"pid"<<getpid()<<std::endl;  for (size\_t i = 0; i < 1000; i++)  {  sleep(1);  }    return 0;  } |

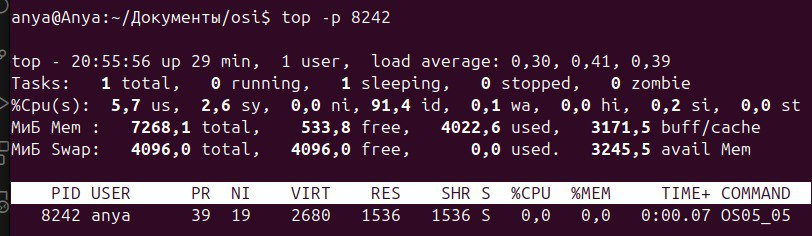












Ответьте на следующие вопросы

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()