Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №5**

Выполнил:

Студент 3 курса 4 группы ФИТ

Сосновец Мария Игоревна

2024 г.

**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;

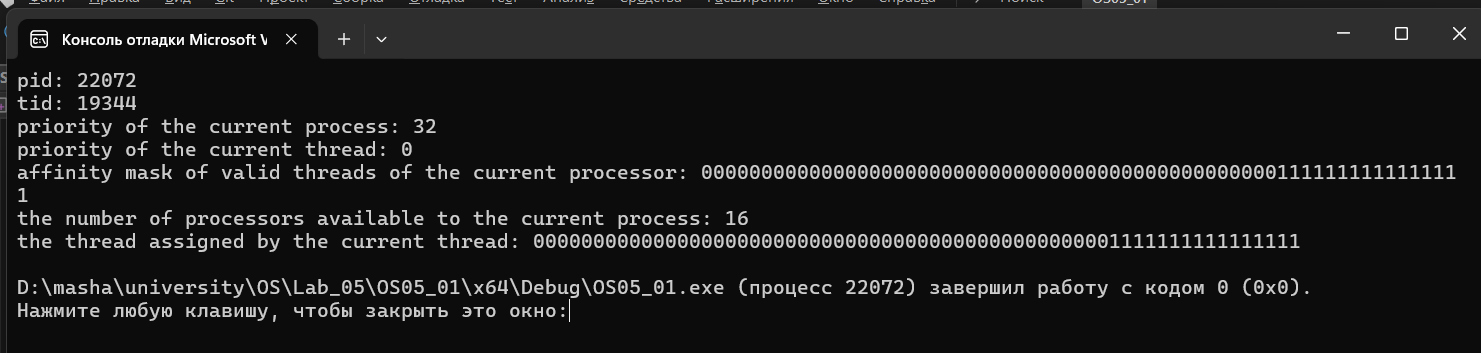
- приоритет текущего потока;

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

- количество процессоров, доступных процессу;

- процессор, назначенный текущему потоку.

|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  void printBinaryMask(DWORD\_PTR mask, int numBits) {  for (int i = numBits - 1; i >= 0; --i) {  cout << ((mask & (1ull << i)) ? '1' : '0');  }  cout << endl;  }  int main() {  setlocale(LC\_ALL, "rus");  DWORD pid = GetCurrentProcessId();  cout << "pid: " << pid << endl;  DWORD tid = GetCurrentThreadId();  cout << "tid: " << tid << endl;  int processPriority = GetPriorityClass(GetCurrentProcess());  cout << "priority of the current process: " << processPriority << endl;  int threadPriority = GetThreadPriority(GetCurrentThread());  cout << "priority of the current thread: " << threadPriority << endl;  DWORD\_PTR processAffinityMask, systemAffinityMask;  GetProcessAffinityMask(GetCurrentProcess(), &processAffinityMask, &systemAffinityMask);  int numProcessors = 0;  for (int i = 0; i < sizeof(DWORD\_PTR) \* 8; i++) {  if (processAffinityMask & (1ull << i)) {  numProcessors++;  }  }  cout << "affinity mask of valid threads of the current processor: ";  printBinaryMask(processAffinityMask, sizeof(DWORD\_PTR) \* 8);  cout << "the number of processors available to the current process: " << numProcessors << endl;  DWORD\_PTR threadAffinityMask = SetThreadAffinityMask(GetCurrentThread(), 1ull << 0);  cout << "the thread assigned by the current thread: ";  printBinaryMask(threadAffinityMask, sizeof(DWORD\_PTR) \* 8);  return 0;  } |

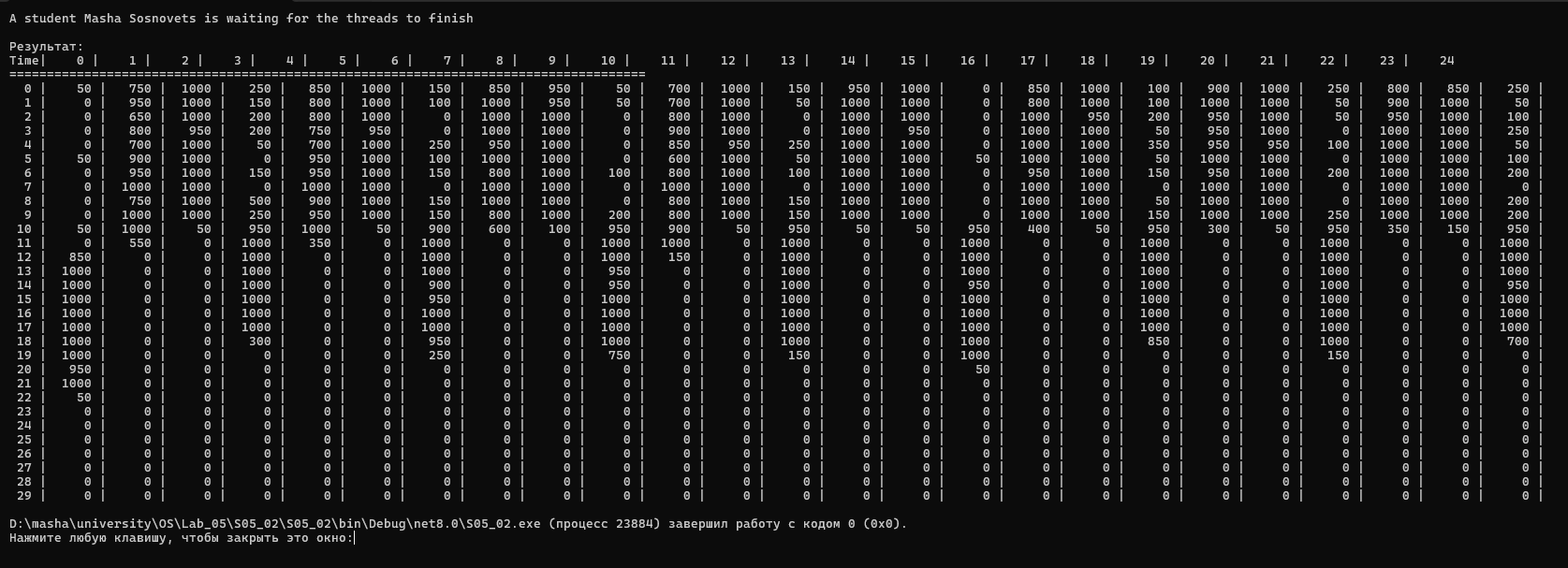


**Задание 02**

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). Как пример, можно использовать следующий фрагмент кода.

|  |
| --- |
| using System;  using System.Diagnostics;  using System.Linq;  using System.Threading;  class Program  {  const int ThreadCount = 25;  const int ThreadLifeTime = 10;  const int ObservationTime = 30;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime;  static void WorkThread(object o)  {  int id = (int)o;  ProcessThread currentThread = Process.GetCurrentProcess().Threads[id];  currentThread.ProcessorAffinity = (IntPtr)(1 << (id % 12));  for (int i = 0; i < ThreadLifeTime \* 20; i++)  {  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)(CurrentTime - StartTime).TotalSeconds;  if (ElapsedSeconds >= 0 && ElapsedSeconds < ObservationTime)  {  Matrix[id, ElapsedSeconds] += 50;  }  MySleep(50);  }  }  static void Main(string[] args)  {  StartTime = DateTime.Now;  Thread[] threads = new Thread[ThreadCount];  for (int i = 0; i < ThreadCount; ++i)  {  threads[i] = new Thread(WorkThread);  if (i % 3 == 0)  {  threads[i].Priority = ThreadPriority.Lowest;  }  else if (i % 3 == 2)  {  threads[i].Priority = ThreadPriority.Highest;  }  else  {  threads[i].Priority = ThreadPriority.Normal;  }  threads[i].Start(i);  }  Console.WriteLine("A student Masha Sosnovets is waiting for the threads to finish");  for (int i = 0; i < ThreadCount; ++i)  threads[i].Join();  Console.WriteLine("\nРезультат:");  Console.WriteLine("Time| " + string.Join(" | ", Enumerable.Range(0, ThreadCount).Select(i => $" {i}")));  Console.WriteLine(new string('=', 85));  for (int s = 0; s < ObservationTime; s++)  {  Console.Write($"{s,3} | ");  for (int th = 0; th < ThreadCount; th++)  {  Console.Write($"{Matrix[th, s],5} | ");  }  Console.WriteLine();  }  }  static void MySleep(int milliseconds)  {  DateTime endTime = DateTime.Now.AddMilliseconds(milliseconds);  while (DateTime.Now < endTime)  {  ComputeFibonacci(20);  }  }  static long ComputeFibonacci(int n)  {  if (n <= 1)  return n;  return ComputeFibonacci(n - 1) + ComputeFibonacci(n - 2);  }  } |

3. Выполните приложение < >, не забудьте про персонификацию вывода результатов.

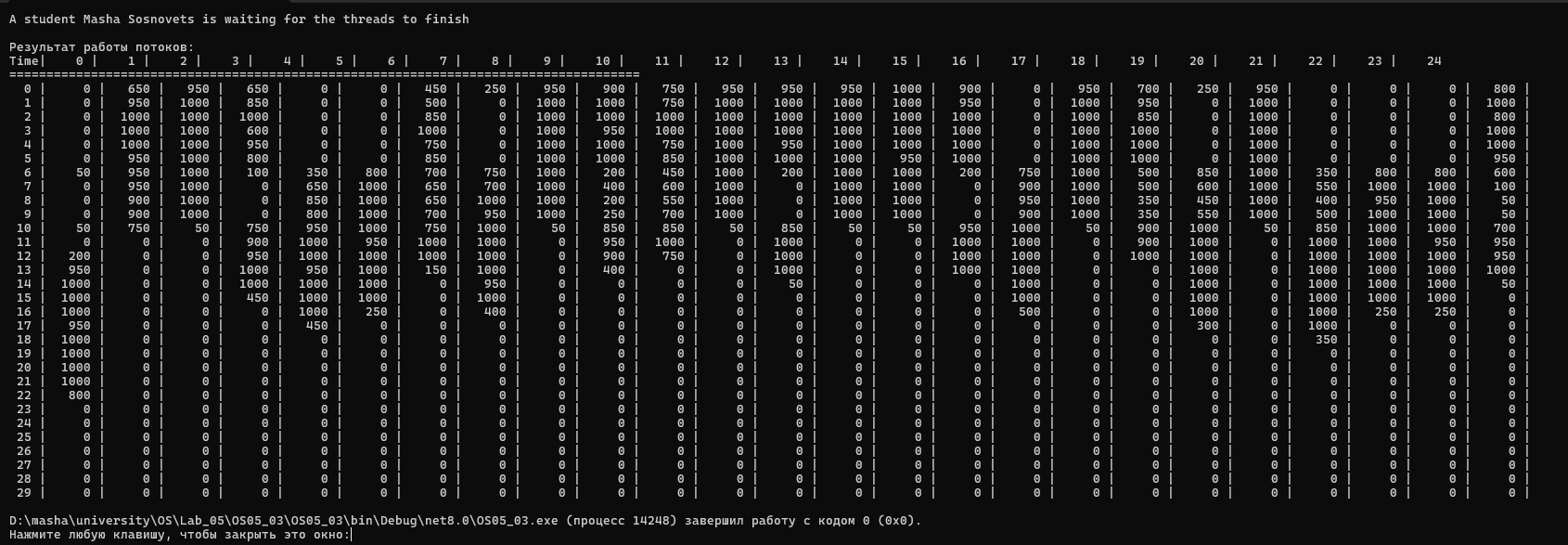


**Задание 03**

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.

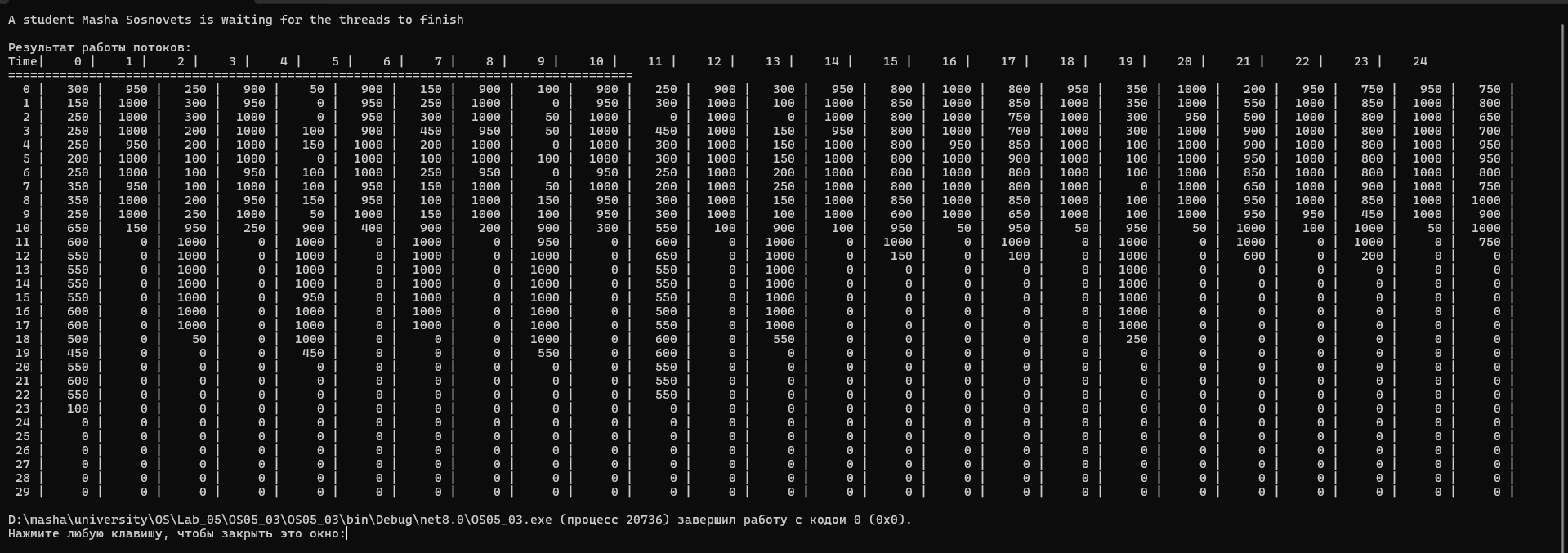
|  |
| --- |
| using System;  using System.Diagnostics;  using System.Linq;  using System.Threading;  class Program  {  const int ThreadCount = 25;  const int ThreadLifeTime = 10;  const int ObservationTime = 30;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime;  static void WorkThread(object o)  {  int id = (int)o;  ProcessThread currentThread = Process.GetCurrentProcess().Threads[id];  currentThread.ProcessorAffinity = (IntPtr)(1 << (id % 12));  for (int i = 0; i < ThreadLifeTime \* 20; i++)  {  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)(CurrentTime - StartTime).TotalSeconds;  if (ElapsedSeconds >= 0 && ElapsedSeconds < ObservationTime)  {  Matrix[id, ElapsedSeconds] += 50;  }  MySleep(50);  }  }  static void Main(string[] args)  {  StartTime = DateTime.Now;  Thread[] threads = new Thread[ThreadCount];  for (int i = 0; i < ThreadCount; ++i)  {  threads[i] = new Thread(WorkThread);  switch (i % 3)  {  case 0:  threads[i].Priority = ThreadPriority.Lowest;  break;  case 2:  threads[i].Priority = ThreadPriority.Highest;  break;  }  threads[i].Start(i);  }  Console.WriteLine("A student Masha Sosnovets is waiting for the threads to finish");  for (int i = 0; i < ThreadCount; ++i)  threads[i].Join();  Console.WriteLine("\nРезультат работы потоков:");  Console.WriteLine("Time| " + string.Join(" | ", Enumerable.Range(0, ThreadCount).Select(i => $" {i}")));  Console.WriteLine(new string('=', 85));  for (int s = 0; s < ObservationTime; s++)  {  Console.Write($"{s,3} | ");  for (int th = 0; th < ThreadCount; th++)  {  Console.Write($"{Matrix[th, s],5} | ");  }  Console.WriteLine();  }  }  static void MySleep(int milliseconds)  {  DateTime endTime = DateTime.Now.AddMilliseconds(milliseconds);  while (DateTime.Now < endTime)  {  ComputeFibonacci(20);  }  }  static long ComputeFibonacci(int n)  {  if (n <= 1)  return n;  return ComputeFibonacci(n - 1) + ComputeFibonacci(n - 2);  }  } |

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?.

|  |
| --- |
| if (i < 2)  t[i].Priority = ThreadPriority.BelowNormal;  else  t[i].Priority = ThreadPriority.Normal; |



8. По зафиксированным скриншотам объясните полученные результаты. При этом укажите числовые значения приоритетов потоков.

Lowest имеет значение приоритета 0

BelowNormal имеет значение приоритета 1

Normal имеет значение приоритета 2

Highest имеет значение приоритета 4

**Задание 04**

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

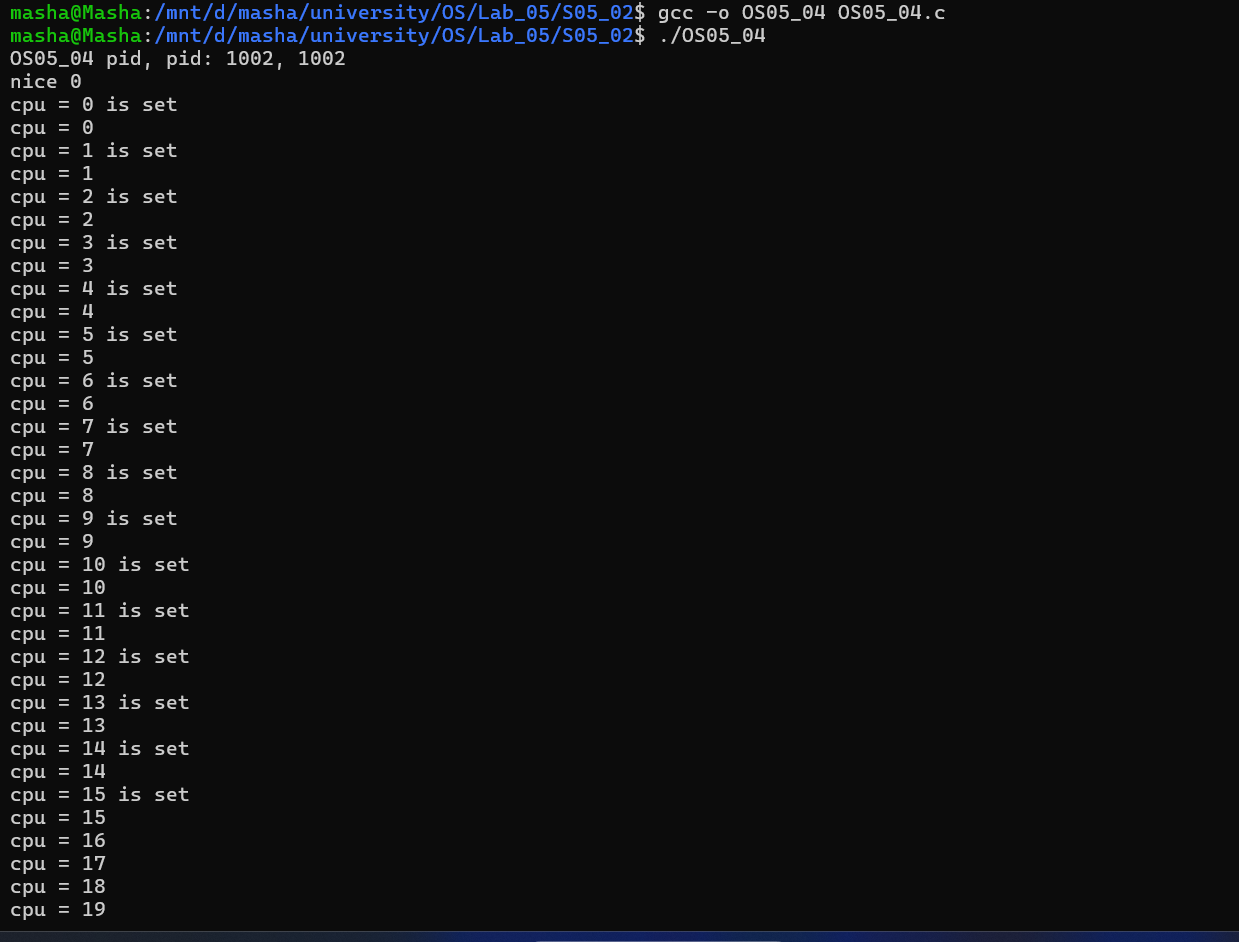
- идентификатор текущего процесса;

- идентификатор текущего (main) потока;

- приоритет (nice) текущего потока;

- номера доступных процессоров.

|  |
| --- |
| #define \_GNU\_SOURCE  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <errno.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <pthread.h>  #include <sys/syscall.h>  #include <sched.h>  #include <sys/syscall.h>  int main() {  pid\_t tid = syscall(SYS\_gettid);  pid\_t pid = getpid();  int cnice = nice(0);  printf("OS05\_04 pid, pid: %d, %d\n", pid, tid);  printf("nice %d\n", cnice);  cpu\_set\_t set;  CPU\_ZERO(&set);  if(sched\_getaffinity(0, sizeof(cpu\_set\_t), &set) == 0) {  int is = 0;  for(int i = 0; i < CPU\_SETSIZE; i++) {  is = CPU\_ISSET(i, &set);  if(is) printf("cpu = %i is %s\n", i, "set");  printf("cpu = %i \n", i);  }  }  else printf("sched\_getaffinity error");    exit(0);  } |

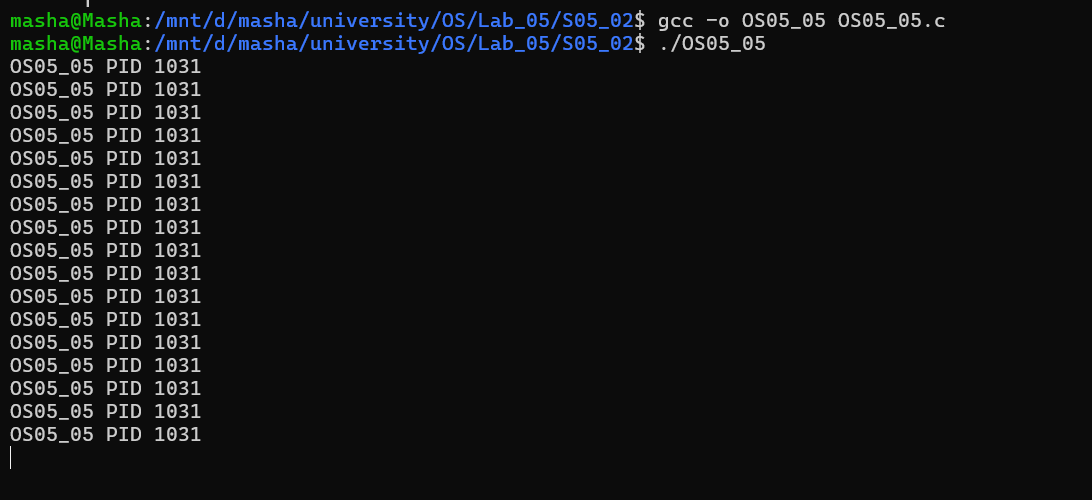


**Задание 05**

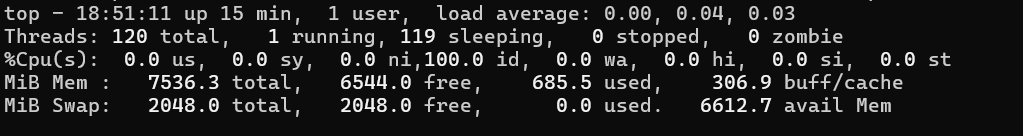
10. Разработайте консольное Linux-приложение OS05\_05 на языке С, выполняющее длинный цикл.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <errno.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <pthread.h>  #include <sys/syscall.h>  #include <sched.h>  int main() {  pid\_t pid = getpid();  nice(0);  for(int i = 0; i < 1000000000; i++) {  printf("OS05\_05 PID %d\n", pid);  sleep(2);  }    exit(0);  } |

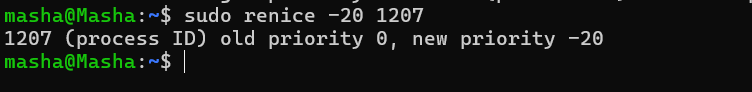
11. Запустите приложение OS05\_05.

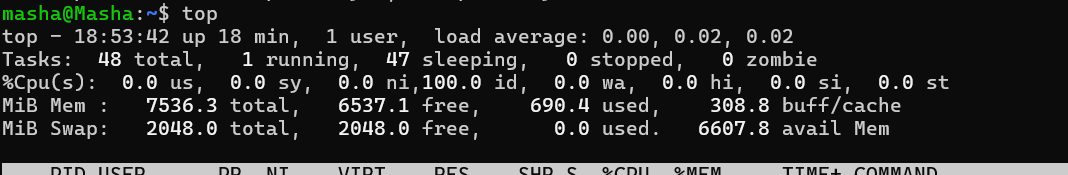


12. Зафиксируйте < > текущее значение nicе, полученное с помощью команды top.

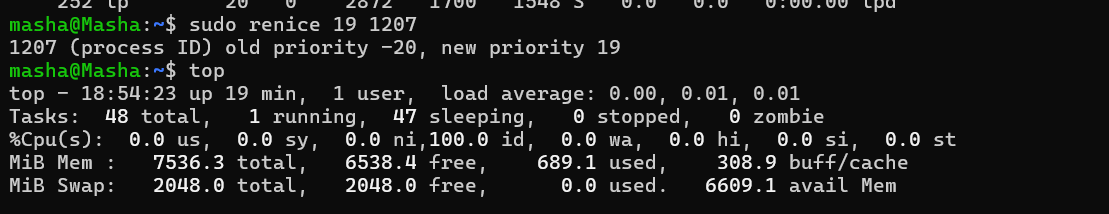


13. Увеличьте приоритет для OS05\_05 до максимального значения (самого привилегированного). Зафиксируйте < > текущее значение nicе, полученное с помощью команды top.





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



4