

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования   
«Санкт-Петербургский государственный технологический институт  
(технический университет)»  
СПбГТИ(ТУ)

|  |  |  |
| --- | --- | --- |
| УГНС | 09.00.00 | Информатика и вычислительная техника |
| Направление подготовки | 09.03.03 | Прикладная информатика |
| Направленность (профиль) |  | Прикладная информатика в химии |
| Факультет |  | Информационных технологий и управления |
| Кафедра |  | Систем автоматизированного  проектирования и управления |
| Учебная дисциплина |  | **ОПЕРАЦИОННЫЕ СИСТЕМЫ** |

Курс 2 Группа 485

**Отчет по лабораторной работе № 4**

|  |  |
| --- | --- |
| **Тема:** | **Управление процессами** |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (инициалы, фамилия)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Макарук Р.В.

(подпись, дата) (инициалы, фамилия)

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись преподавателя)

Санкт-Петербург

2020

**1. Цель работы**

Получение навыков управления штатными средства ОС для исследования процессов. Также требуется разработать программу, реализующую алгоритм банкира

**Исходные данные для тестирования алгоритма банкира**

В контексте «алгоритма банкира» определите и обоснуйте, является ли приведенное состояние опасным или безопасным с точки зрения возникновения тупиков.

Предположим, что в системе имеются 4 одинаковых ресурсов R1, 4 одинаковых ресурсов R2, 4 одинаковых ресурсов R3 и 4 одинаковых ресурсов R4. Текущее распределение ресурсов и максимальное их количество, необходимое процессам представлено в таблице 1.

Таблица 1 – Распределение ресурсов

|  |  |  |
| --- | --- | --- |
| **Процесс** | **Предоставлено ресурсов**  **R1 R2 R3 R4** | **Максимальная потребность**  **R1 R2 R3 R4** |
| A | 2 0 0 0 | 2 0 2 2 |
| B | 2 2 0 0 | 2 2 2 2 |
| C | 0 2 2 0 | 2 4 2 4 |
| D | 0 0 2 2 | 0 0 2 4 |

**2. Ход работы**

Код программы:

[файл Program.cs]

using System;

using System.Collections.Generic;

using System.Linq;

namespace Lab4

{

class Program

{

static readonly string NL = Environment.NewLine;

static void OutputResourcesNames(List<Resource> resources)

{

Console.Write(" ");

foreach (var res in resources)

Console.Write(res.Name);

Console.WriteLine();

}

static void OutputProcessesWithGivenRes(List<Process> processes)

{

foreach (var process in processes)

{

Console.Write(process.Name + " ");

foreach (var resource in process.GivenResources)

Console.Write(resource.Val + " ");

Console.WriteLine();

}

Console.WriteLine();

}

static void OutputProcessesNeededRes(List<Process> processes)

{

foreach (var process in processes)

{

Console.Write(process.Name + " ");

foreach (var resource in process.NeededResources)

Console.Write(resource.Val + " ");

Console.WriteLine();

}

}

static void OutputResourcesVector(List<Resource> resources)

{

Console.Write("{ ");

for (int i = 0; i < resources.Count; ++i)

{

Console.Write(resources[i].Val);

if (i != resources.Count - 1)

Console.Write(", ");

}

Console.WriteLine(" }");

}

static bool BankerAlgorithm(List<Process> processes, List<Resource> availableResources)

{

Console.WriteLine("Алгоритм банкира:" + NL);

bool safeCondition = true;

while (processes.Any() && safeCondition)

{

Console.WriteLine("Предоставлено ресурсов:");

OutputResourcesNames(availableResources);

OutputProcessesWithGivenRes(processes);

Console.WriteLine("Максимальная потребность:");

OutputResourcesNames(availableResources);

OutputProcessesNeededRes(processes);

Console.Write("Ресурсов осталось:");

OutputResourcesVector(availableResources);

int processesNum = processes.Count;

foreach (var process in processes)

{

bool toProvideByResource = true;

foreach (var neededResource in process.NeededResources)

{

var givenRes = process.GivenResources.Find(r => r == neededResource);

var availableRes = availableResources.Find(r => r == neededResource);

if ((availableRes + givenRes) < neededResource)

{

toProvideByResource = false;

break;

}

}

if (toProvideByResource)

{

for (int i = 0; i < availableResources.Count; ++i)

{

var availableResource = availableResources[i];

var givenRes = process.GivenResources.Find(r => r == availableResource);

availableResource.Val += givenRes.Val;

}

Console.WriteLine("Процесс " + process.Name + " был завершён и отдал:");

OutputResourcesVector(process.GivenResources);

Console.WriteLine();

processes.Remove(process);

break;

}

}

if (processes.Count == processesNum)

safeCondition = false;

}

return safeCondition;

}

static void Main()

{

var R1 = new Resource("R1", 4);

var R2 = new Resource("R2", 4);

var R3 = new Resource("R3", 4);

var R4 = new Resource("R4", 4);

var resources = new List<Resource> { R1, R2, R3, R4 };

var A = new Process("A", new List<Resource> { R1 - 2, R2 - 4, R3 - 4, R4 - 4 }, new List<Resource> { R1 - 2, R2 - 4, R3 - 2, R4 - 2 });

var B = new Process("B", new List<Resource> { R1 - 2, R2 - 2, R3 - 4, R4 - 4 }, new List<Resource> { R1 - 2, R2 - 2, R3 - 2, R4 - 2 });

var C = new Process("C", new List<Resource> { R1 - 4, R2 - 2, R3 - 2, R4 - 4 }, new List<Resource> { R1 - 2, R2 - 0, R3 - 2, R4 - 0 });

var D = new Process("D", new List<Resource> { R1 - 4, R2 - 4, R3 - 2, R4 - 2 }, new List<Resource> { R1 - 4, R2 - 4, R3 - 2, R4 - 0 });

var processes = new List<Process> { A, B, C, D };

var availableResources = resources;

foreach (var process in processes)

foreach (var givenRes in process.GivenResources)

{

var res = availableResources.Find(r => r == givenRes);

res.Val -= givenRes.Val;

}

bool safeCondition = BankerAlgorithm(processes, availableResources);

if (safeCondition)

Console.WriteLine("Состояние безопасно.");

else

Console.WriteLine("Состояние небезопасно.");

}

}

}

[конец файла Program.cs]

[файл Process.cs]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab4

{

class Process

{

public string Name { get; private set; }

public List<Resource> GivenResources { get; private set; }

public List<Resource> NeededResources { get; private set; }

public Process(string name, List<Resource> givenResources, List<Resource> neededResources)

{

Name = name;

GivenResources = givenResources;

NeededResources = neededResources;

}

}

}

[конец файла Process.cs]

[файл Resource.cs]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab4

{

public class Resource

{

public string Name { get; set; }

public int Val { get; set; }

public Resource(string name, int val)

{

Name = name;

Val = val;

}

public static Resource operator +(Resource a, Resource b) => new Resource(a.Name, a.Val + b.Val);

public static Resource operator -(Resource a, Resource b) => new Resource(a.Name, a.Val - b.Val);

public static Resource operator +(Resource a, int b) => new Resource(a.Name, a.Val - b);

public static Resource operator -(Resource a, int b) => new Resource(a.Name, a.Val - b);

public static bool operator <(Resource a, Resource b) => a.Val < b.Val;

public static bool operator >(Resource a, Resource b) => a.Val > b.Val;

public static bool operator <=(Resource a, Resource b) => a.Val <= b.Val;

public static bool operator >=(Resource a, Resource b) => a.Val >= b.Val;

public static bool operator ==(Resource left, Resource right)

{

return EqualityComparer<Resource>.Default.Equals(left, right);

}

public static bool operator !=(Resource left, Resource right)

{

return !(left == right);

}

public override bool Equals(object obj)

{

return obj is Resource resource &&

Name == resource.Name;

}

public override int GetHashCode()

{

return 539060726 + EqualityComparer<string>.Default.GetHashCode(Name);

}

}

}

[конец файла Resource.cs]

**2.1 Результат работы программы**

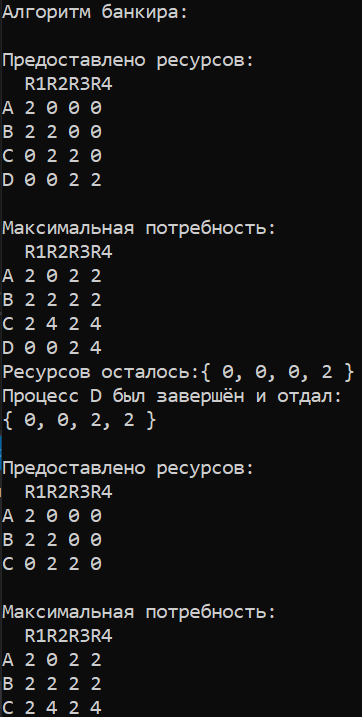


Рисунок 1 – Результат работы программы

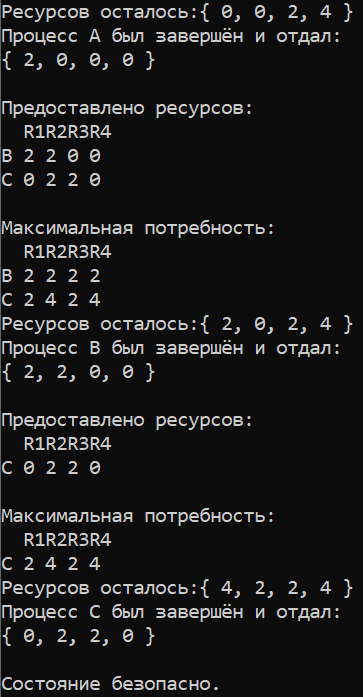


Рисунок 2 – Результат работы программы

**2.2 Штатные средства ОС для исследования процессов**

В качестве тестовой системы для выполнения практического задания использоваласьMS Windows 10.

В качестве исследуемого приложения был выбран проигрыватель VLC. Выполнимая операция – воспроизведение мультимедиа.

В диспетчере задач MS Windows виден созданный процесс (Рисунок 3).

Диспетчер задач позволяет получить обобщенную информацию об использовании основных ресурсов компьютера: общее количество процессов и потоков, участвующих в системе и т.д.

В диспетчере задач отображаются сведения о процессах, выполняемых на компьютере. Кроме того, там можно просмотреть наиболее часто используемые показатели быстродействия процессов.

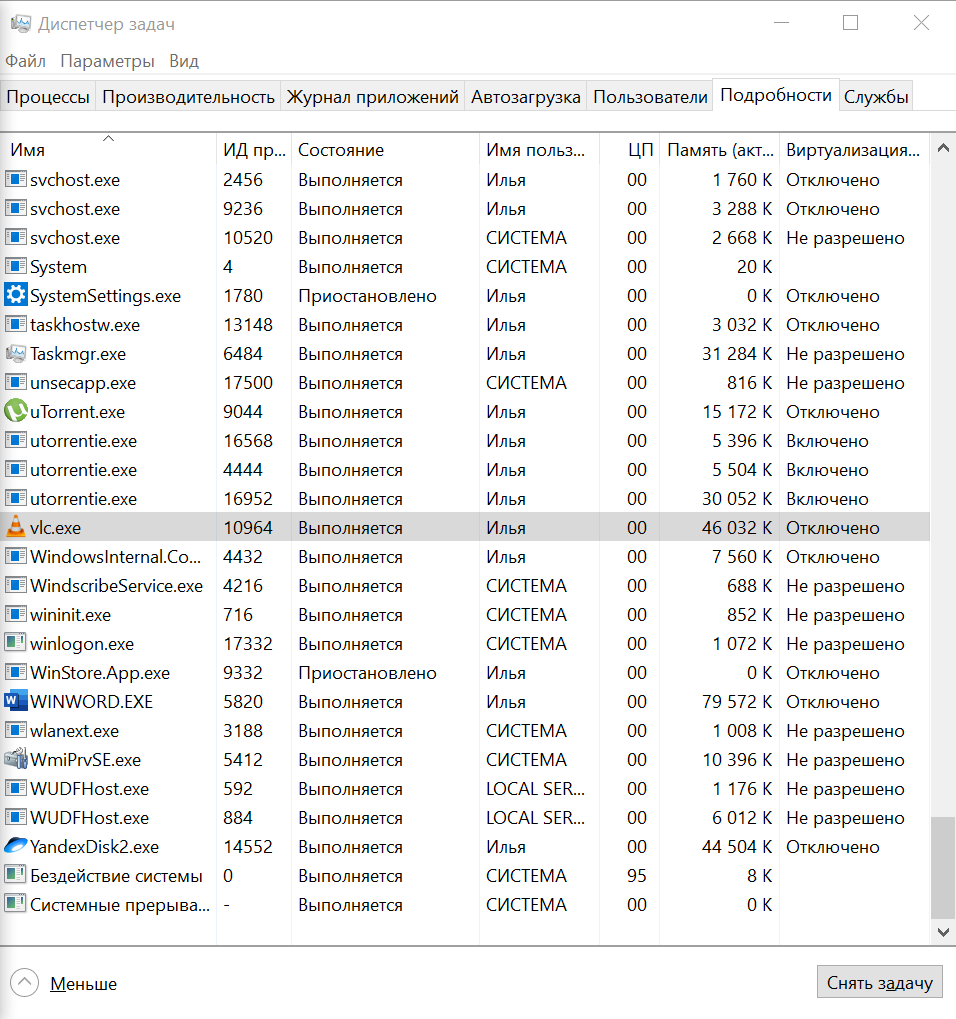


Рисунок 3 – Характеристики запущенного процесса

Просмотр (мониторинг) выполнения процесса со счётчиками представлен на Рисунке 4.

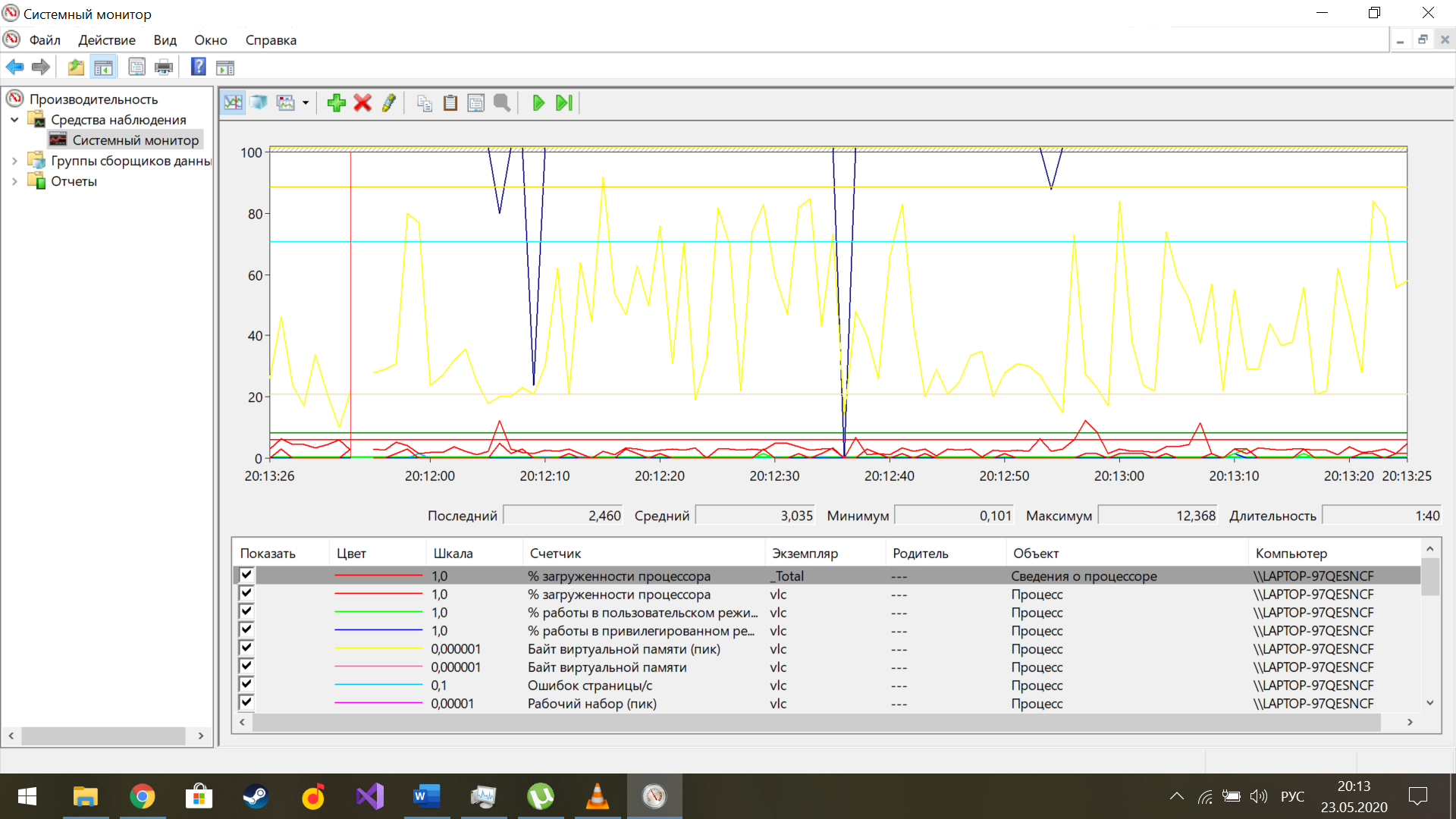


Рисунок 4 – Выполнение операции при среднем приоритете

Системный монитор служит для сбора и просмотра в реальном времени данных памяти, диска, процессора, сети и других параметров в виде графика, гистограммы или отчета.

Анализ данных наблюдения позволяет обнаружить такие явления, как избыточный спрос на определенные ресурсы, приводящий к возникновению узкого места в работе системы.

Перед выполнением исследуемой задачи устанавливаем сначала минимальный приоритет процесса (Рисунок 5)

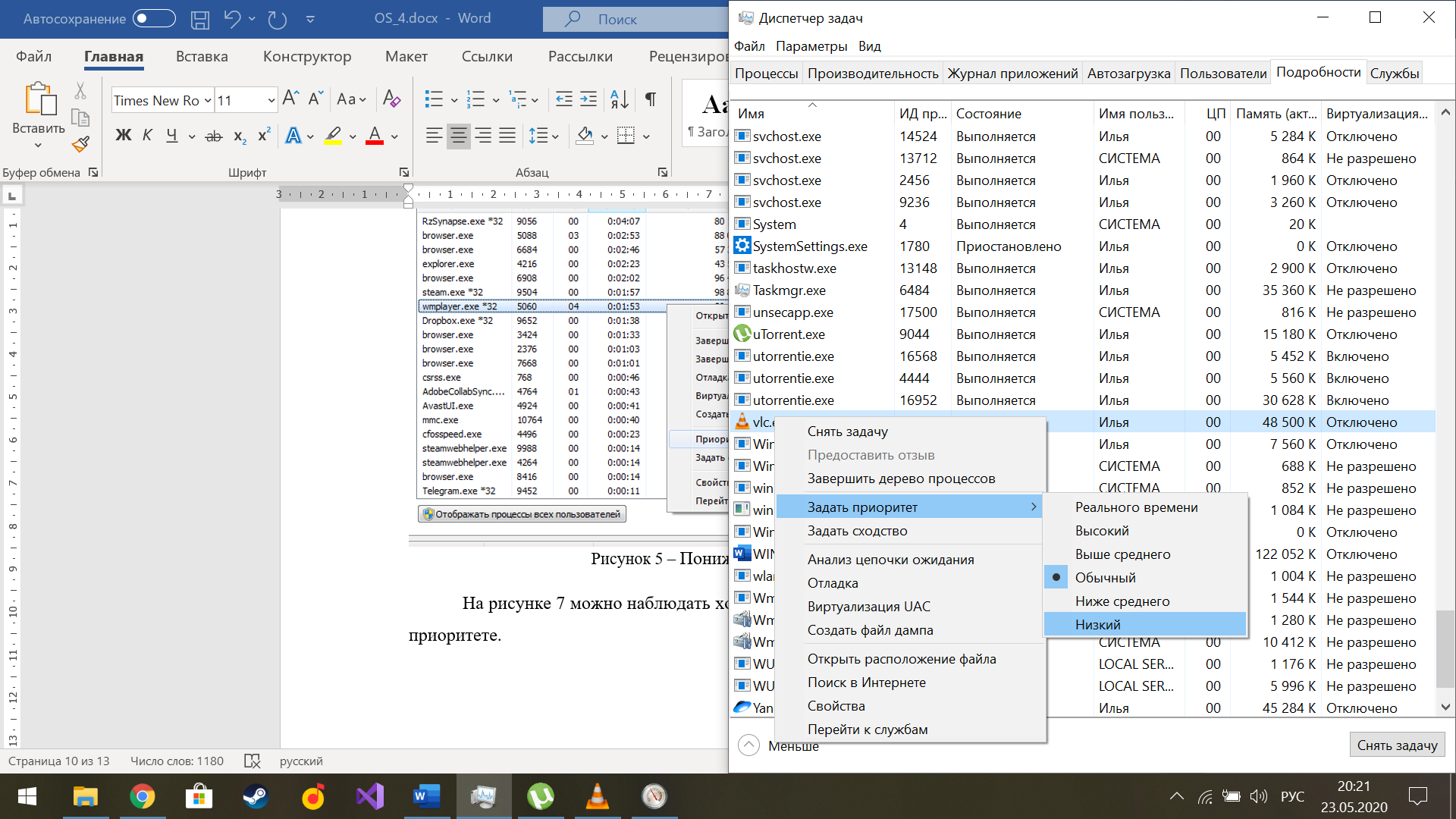


Рисунок 5 – Понижение базового приоритета

На Рисунке 6 можно наблюдать ход выполнения самой емкой операции при низком приоритете.

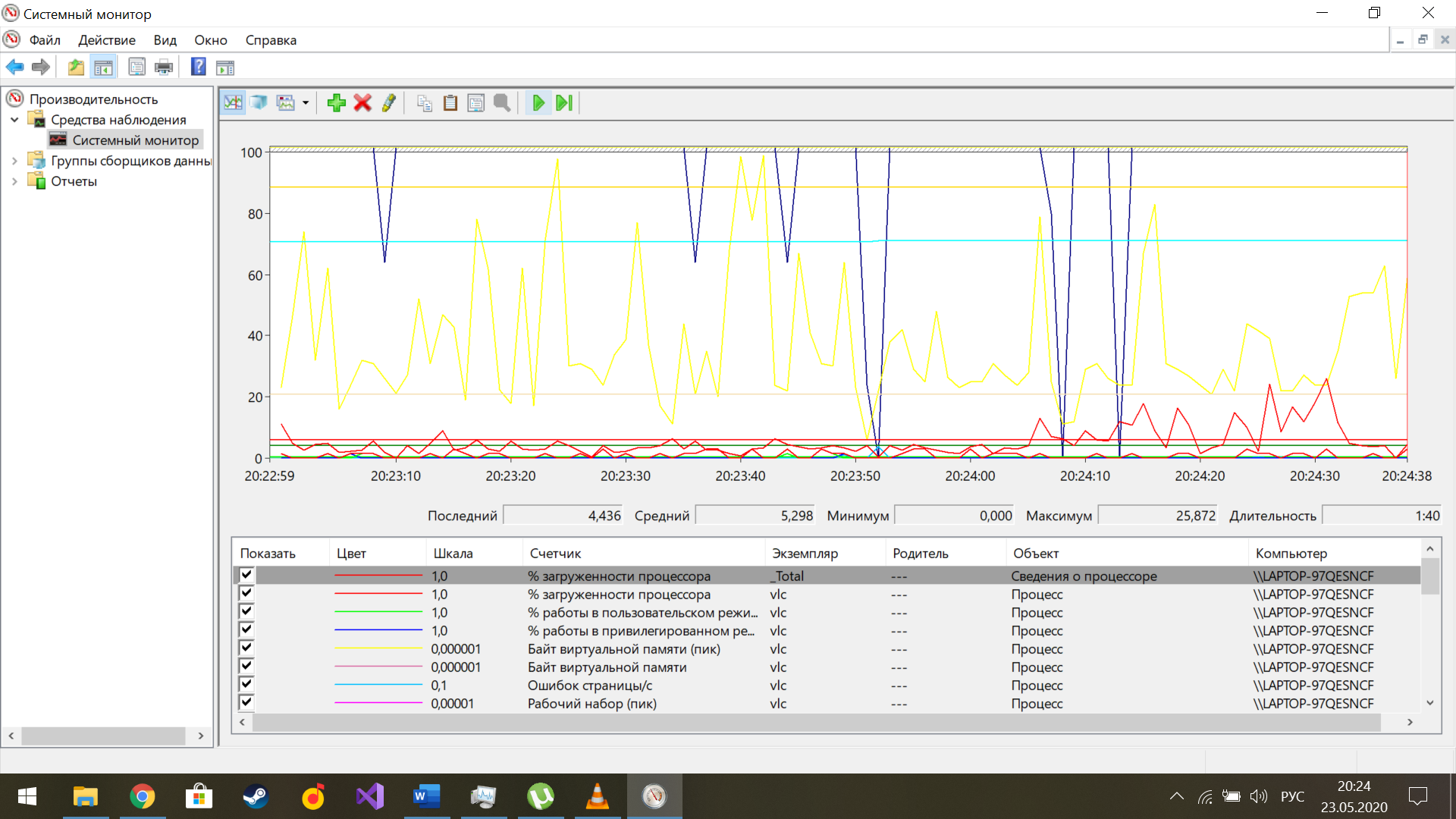


Рисунок 6 – Выполнение операции при пониженном приоритете

Изменим приоритет запущенного процесса (Рисунок 7)

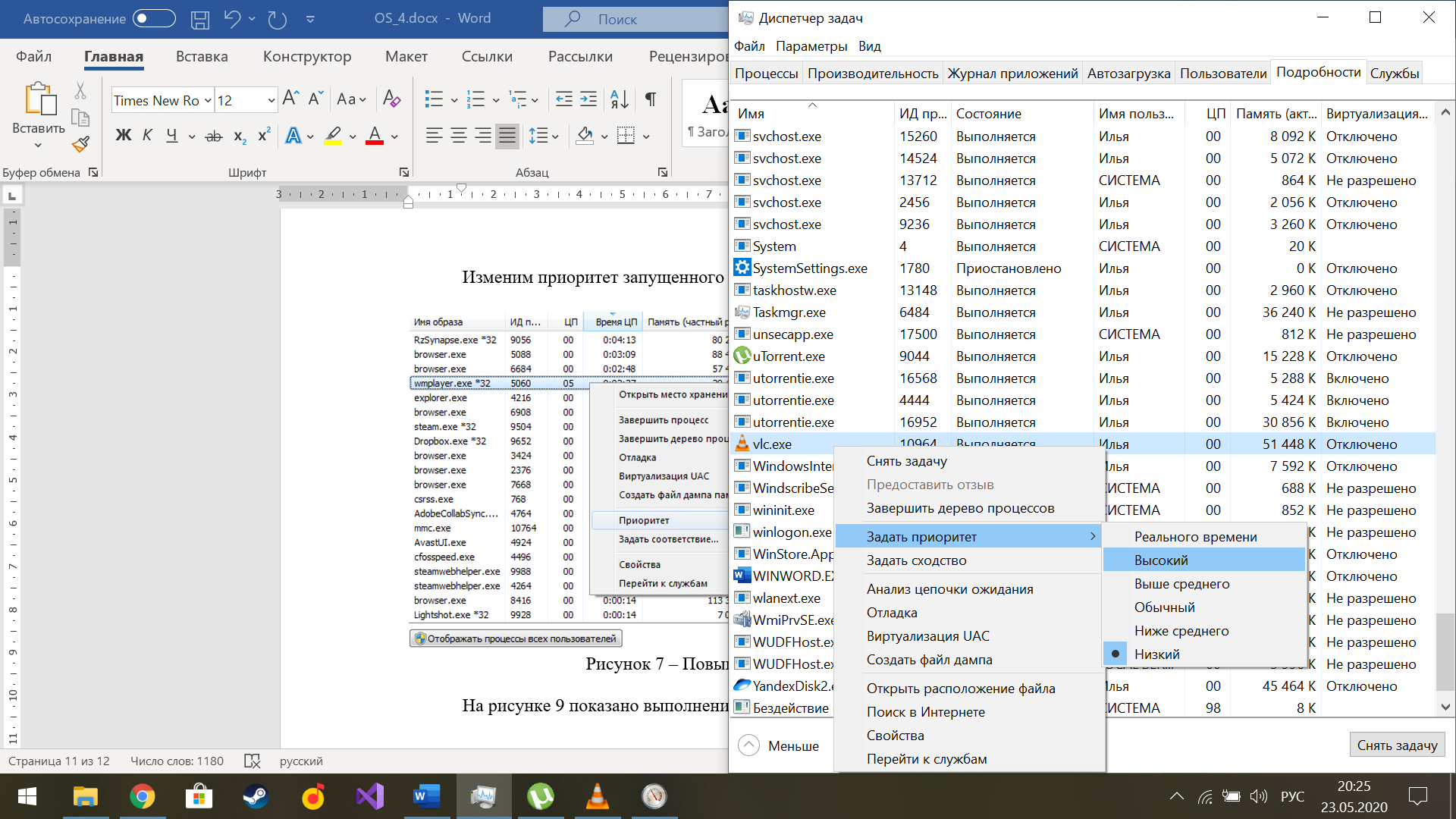


Рисунок 7 – Повышение базового приоритета

На Рисунке 8 показано выполнение той же операции с повышением приоритета.

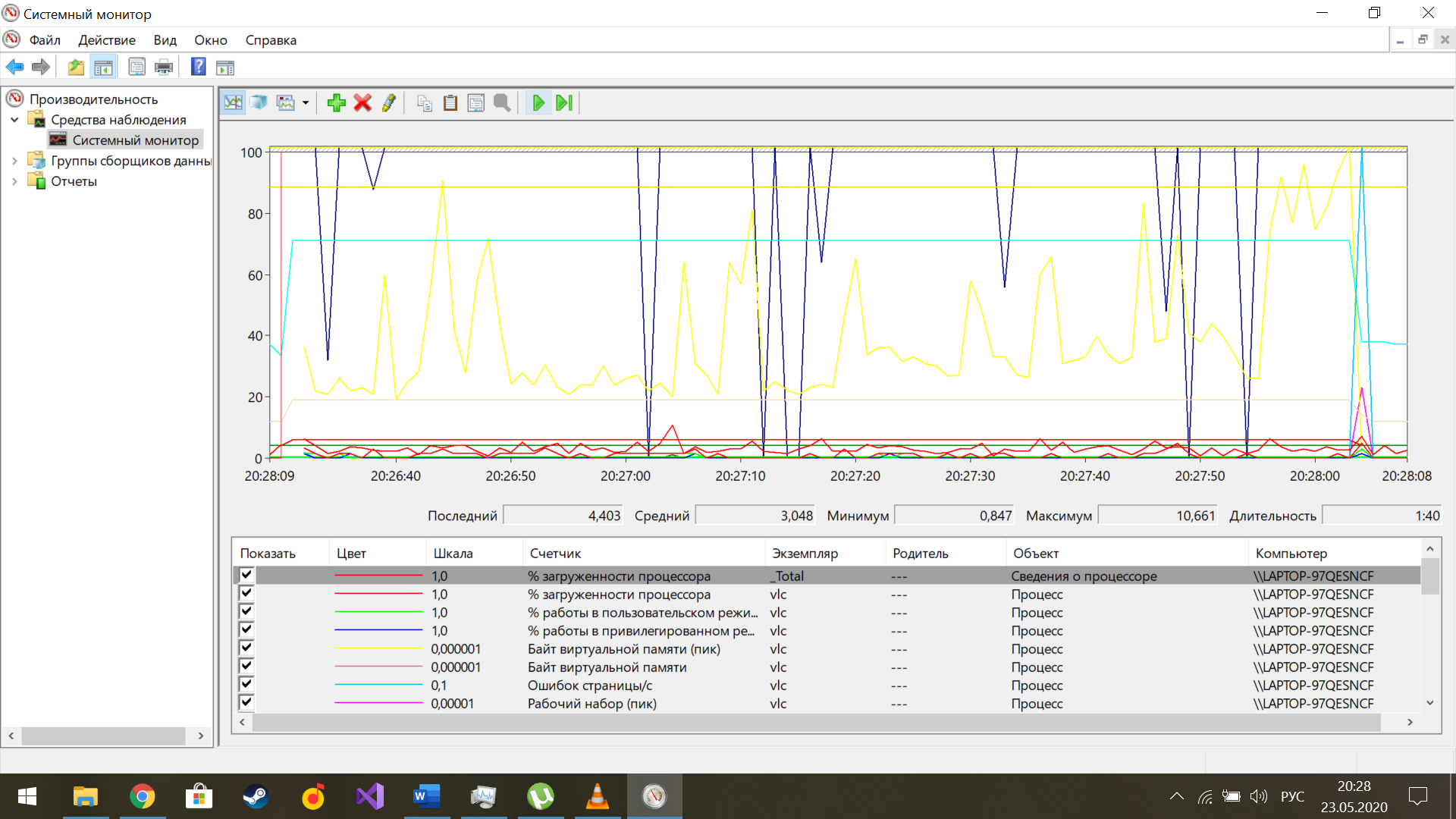


Рисунок 8 – Процесс выполняется с повышенным приоритетом

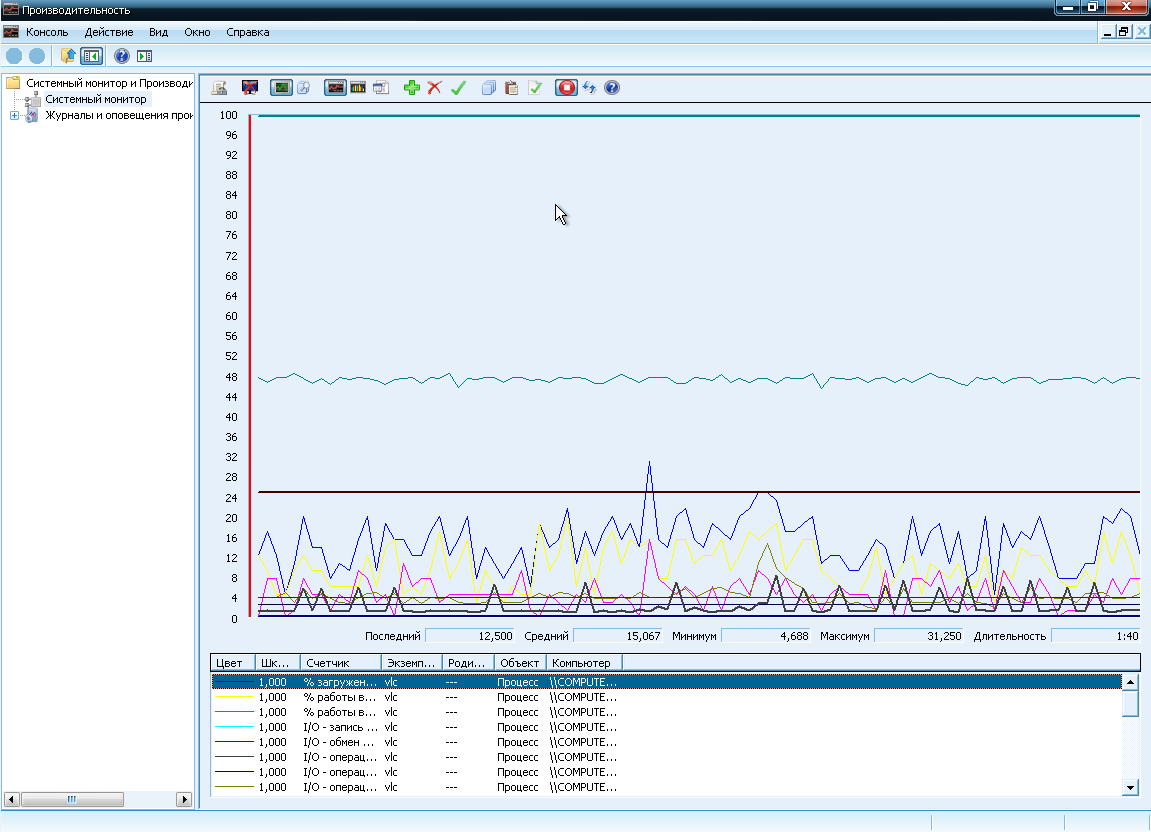
Из представленных рисунков очевидно, что при изменении приоритета процесса резко изменяется характер работы процесса в пользовательском режиме в много процессной системе - задача выполняется быстрее.

Диспетчер задач может показать количество потоков, созданных в конкретном процессе. В рассматриваемом процессе было создано 24 потока. Эта многопоточность в рамках одного процесса положительно влияет на производительность задачи.

Теперь посмотрим на работу VLC в Windows XP. При обычном приоритете Рисунок 9.

Рисунок 9 – Выполнение операции в обычном приоритете.

Теперь понизим приоритет и взглянем на работу (Рисунок 10).

Рисунок 10 – Выполнение операции при пониженном приоритете

Теперь проследим за работой процесса при высоком приоритете (Рисунок 11).

Рисунок 11 – Выполнение операции при повышенном приоритете.

Как и в Windows 10, процессы в Windows XP положительно реагируют на повышение приоритета, система выделяет на их выполнение больше ресурсов, что ускоряет их работу.

**3 Ответы на вопросы**

**Концепция дискретных состояний процесса.**

В период существования процесс проходит через ряд дискретных состояний. Смену состояний могут вызвать различные события. Базовые состояния процессов в системе:

1. Готов (Ready) - находится в состоянии готовности (т.е. может выполняться). Процесс, которому выделены все ресурсы кроме ЦП носит название готового к исполнению. Когда процессу выделяется ЦП - происходит смена состояний. Предоставление процессу ЦП называется запуском. Одновременно в системе может находиться несколько процессов (список готовых к исполнению процессов).
2. Выполняется (Running) - продолжающиеся (в состоянии выполнения). Говорит, что процесс выполняется, если ему выделен ЦП. В одном процессоре только один процесс может находиться в исполнении.
3. Блокирован (Blocked) - процесс блокирован или в состоянии блокирования, если не может выполняться пока не получен необходимый ему ресурс или сообщение от другого процесса.

**Стратегии разрешения тупиков.**

Методы предотвращения тупиков ориентированы главным образом на нарушение первых трех условий возникновения тупиков (условие взаимоисключения, условие ожидания ресурсов и условие неперераспределяемости) путем введения ряда ограничений на поведение процессов и способы распределения ресурсов. Эти методы обнаружения и устранения сводятся к поиску и разрыву цикла ожидания ресурсов.

Основные стратегии борьбы с тупиками:

* Игнорирование проблемы в целом
* Предотвращение тупиков
* Обнаружение тупиков
* Восстановление после тупиков

**4.Вывод**

В ходе выполнения данной работы, на примере MS Windows 10, были получены навыки управления средствами ОС для изучения работы процессов, были отмечены основные стратегии борьбы с тупиками, а также была разработана программа, реализующая алгоритм банкира.