ГУАП

КАФЕДРА № 44

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | Н. В. Кучин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| ОСОБЕННОСТИ ДИСПЕТЧЕРИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ, ГРАФ СОСТОЯНИЙ ПРОЦЕССА |
| по курсу: ОПЕРАЦИОННЫЕ СИСТЕМЫ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4941 |  |  |  | И.В. Зеленов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**Цель работы:** написать и отладить программу, моделирующую работу диспетчера операционной системы.

**Вариант 14**

Диспетчер на основе дисциплины с относительными приоритетами (текущая задача выполняется до конца, а затем начинается выполнение готовой задачи с максимальным приоритетом). Данная дисциплина является невытесняющей и характерна для обработки прерываний.Код программы:

*Листинг 1*

*XAML разметка*

<Window x:Class="Dispatcher14.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Dispatcher14"

mc:Ignorable="d"

Title="DispatcherWindow" Height="450" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="200"/>

<RowDefinition/>

</Grid.RowDefinitions>

<ScrollViewer Margin="5" Grid.Row="0" VerticalScrollBarVisibility="Disabled" HorizontalScrollBarVisibility="Visible">

<Grid x:Name="MyGrid">

<Grid Margin="0 0 0 30" VerticalAlignment="Center" HorizontalAlignment="Left">

<StackPanel HorizontalAlignment="left"

VerticalAlignment="Center"

Orientation="Horizontal"

Margin="5 0 0 0"

x:Name="ActiveBar"

>

</StackPanel >

</Grid>

<Grid Margin="0 90 0 0" VerticalAlignment="Top" HorizontalAlignment="Left">

<StackPanel HorizontalAlignment="left"

VerticalAlignment="Center"

Orientation="Horizontal"

Margin="5 0 0 0"

x:Name="LoadBar"

>

</StackPanel >

</Grid>

</Grid>

</ScrollViewer>

<DockPanel Grid.Row="1">

<Button Content="Запуск"

Width="100"

Height="30"

DockPanel.Dock="Right"

Click="Start\_Click"/>

<TextBox x:Name="data"

Margin="5"

BorderThickness="3"

BorderBrush="Black"

DockPanel.Dock="Left"/>

</DockPanel>

</Grid>

</Window>

Листинг 2

Класс MainWindow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Dispatcher14

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

static int pxLen = 60; //длина такта на схеме

static string Path;

public MainWindow()

{

InitializeComponent();

}

public void createLine(int length) //генерация оси тактов

{

int lineLength = (length + 1) \* pxLen;

Border VerticalLine = new Border { Background = Brushes.Black, BorderBrush = Brushes.Black, BorderThickness = new Thickness(2), Width = lineLength, Height = 5, Margin = new Thickness(0, 0, 0, 0), HorizontalAlignment = HorizontalAlignment.Left, VerticalAlignment = VerticalAlignment.Center };

this.MyGrid.Children.Add(VerticalLine);

for (int i = 0; i < length; i++)

{

int marg = i \* pxLen;

Border border = new Border { Background = Brushes.Black, BorderBrush = Brushes.Black, BorderThickness = new Thickness(2), Width = 5, Height = 40, Margin = new Thickness(marg, 0, 0, 0), HorizontalAlignment = HorizontalAlignment.Left, VerticalAlignment = VerticalAlignment.Center };

MyGrid.Children.Add(border);

}

}

void Do()

{

//парсим входную строку

List<MyTask> Tasks = new List<MyTask>();

string textData = data.Text;

var DataList = textData.Split('/');

foreach (var str in DataList)

{

var task = str.Split(' ');

Tasks.Add(new MyTask(task[0], Convert.ToInt32(task[1]), Convert.ToInt32(task[2]), Convert.ToInt32(task[3])));

}

//теперь в списке лежат все задачи

//время их обработать

int i = 0; //счётчик тактов

MyTask lastActive = null; //прошлый процесс

while (Tasks.Count() != 0)

{

var readyTasks = Tasks.Where(x => x.activeTime <= i).ToList(); //выбираем из процессов готовые

if (lastActive != null) //если он не пуст

{

lastActive.work++;

addActive(lastActive); // добавляем отображдение в линию активных процессов

readyTasks.Remove(lastActive);

if (lastActive.work >= lastActive.time) //закончился?

{

Tasks.Remove(lastActive); //удалить из списка задач

lastActive = null;

}

}

else

{

var activeTask = readyTasks.FirstOrDefault(x => x.Pr == readyTasks.Max(y => y.Pr)); //один активный процесс

if (activeTask ==null)

{

activeTask = new MyTask(" ", 0, 0, 0);

}

else

{

activeTask.work++;// отработал такт

readyTasks.Remove(activeTask);

if (activeTask.work >= activeTask.time)//закончился?

{

Tasks.Remove(activeTask);//удалить из списка задач

}

else

{

lastActive = activeTask;

}

}

addActive(activeTask); // добавляем отображдение в линию активных процессов

}

addLoaded(readyTasks);// добавляем отображдение в линию готовых процессов

i++;

}

createLine(i);

}

private void Start\_Click(object sender, RoutedEventArgs e)//Ораотчик кнопки

{

LoadBar.Children.Clear();

ActiveBar.Children.Clear();

Do();

}

private void addActive(MyTask task)

{

TaskBox Box = new TaskBox(); // добавляем отображдение в линию активных процессов

Box.Title = task.name;

Box.BackGround = Brushes.Green;

Box.Width = 55;

Box.Margin = new Thickness(0, 0, 5, 0);

ActiveBar.Children.Add(Box);

}

private void addLoaded(List<MyTask> tasks)

{

StackPanel stackPanel = new StackPanel();// добавляем отображдение в линию готовых процессов

stackPanel.Orientation = Orientation.Vertical;

stackPanel.Margin = new Thickness(0, 0, 5, 0);

if (tasks.Count() == 0)

{

TaskBox taskBox = new TaskBox();

taskBox.Content = "";

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

foreach (MyTask task in tasks)

{

if (task.name != "null")

{

TaskBox taskBox = new TaskBox();

taskBox.BackGround = Brushes.Gray;

taskBox.Title = task.name;

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

}

LoadBar.Children.Add(stackPanel);

}

}

public class MyTask

{

public string name;

public int time; //время исполнения

public int activeTime; //время активациии процесса

public int Pr; //приоритет

public int work; //сколько уже выполнялся

public MyTask(string Name, int Time, int AciveTime, int Pr)

{

name = Name;

time = Time;

activeTime = AciveTime;

this.Pr = Pr;

work = 0;

}

}

}

**Скриншоты работы программы**



Рисунок 1 – Тестовые значения

(имя, время обработки, время выполнения, приоритет)

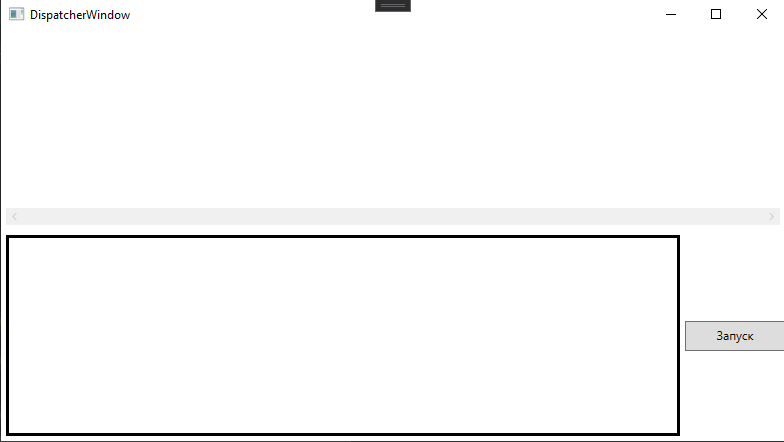


Рисунок 2 – начальное окно

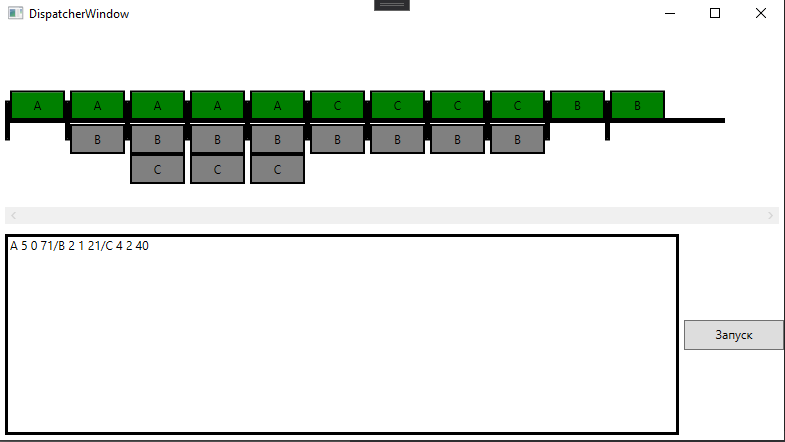


Рисунок 3 – результат расчёта