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 |  |  |  | М.П. Леднёв |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

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

**Вариант 17**

Диспетчер на основе дисциплины SRT.

SRT (shortest remaining time) – прежде процессор получает та задача, которая имеет меньше всего времени для своего завершения. Это время определяется как разность между заказанным временем обслуживания и тем процессорным временем, которая задача уже получила. SRT свободна от недостатка, характерного для SJN. SRT сложна в реализации и дискриминационна по отношению к “длинным” процессам.

Листинг 1. MainWindow.xaml

<Window x:Class="Dispatcher.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:Dispatcher"

mc:Ignorable="d"

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

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<DockPanel>

<StackPanel DockPanel.Dock="Top">

<TextBlock Text="Орабатывать раз в " VerticalAlignment="Top" HorizontalAlignment="Left"/>

<StackPanel Orientation="Horizontal">

<TextBox x:Name="Time" Text="1" Margin="0 5 0 0" Height="20" Width="20" HorizontalAlignment="Left" VerticalAlignment="Top"/>

<TextBlock Text="тактов" Margin="5 5 0 0" />

</StackPanel>

</StackPanel>

<Button Content="Запустить"

Height="25"

Width="100"

Click="Button\_Click"

DockPanel.Dock="Bottom"/>

</DockPanel>

<Grid Margin="5"

Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="150"/>

</Grid.RowDefinitions>

<ScrollViewer VerticalScrollBarVisibility="Disabled"

HorizontalScrollBarVisibility="Visible"

HorizontalAlignment="Center">

<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="0 140 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>

<Border Background="Gray"

BorderBrush="Beige"

BorderThickness="5"

Grid.Row="1">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<TextBox x:Name="source"

DockPanel.Dock="Left"

Background="Transparent"

BorderBrush="DarkGray"

BorderThickness="2"

Cursor="Arrow"

PreviewDragOver="txtTarget\_DropOver"

AllowDrop="true"

Drop="txtTarget\_Drop"

/>

<StackPanel DockPanel.Dock="Right"

HorizontalAlignment="Right"

Grid.Column="1"

Margin="5">

<Button Content="Сохранить"

Height="25"

Width="100"

Margin="0 0 0 5"

Click="Save\_Click"/>

<Button Content="Загрузить"

Height="25"

Width="100"

Margin="0 0 0 5"

Click="Load\_Click"/>

</StackPanel>

</Grid>

</Border>

</Grid>

</Grid>

</Window>

Листинг 2. MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.IO;

using Microsoft.Win32;

namespace Dispatcher

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

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

public MainWindow()

{

InitializeComponent();

}

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

{

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

Border VerticalLine = new Border { Background = Brushes.Wheat, 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.Wheat, 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);

}

}

private void txtTarget\_Drop(object sender, DragEventArgs e) //оработка Drag&Drop

{

string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);

using (StreamReader sr = new StreamReader(files[0]))

{

(sender as TextBox).Text = sr.ReadToEnd();

}

}

private void txtTarget\_DropOver(object sender, DragEventArgs e) //оработка Drag&Drop

{

e.Handled = true;

}

private void Button\_Click(object sender, RoutedEventArgs e)//Обработчик кнопки рассчёт

{

LoadBar.Children.Clear();

ActiveBar.Children.Clear();

solve(Convert.ToInt32(Time.Text));

}

private void Load\_Click(object sender, RoutedEventArgs e)//Обработчик кнопки загрузить

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

source.Text = File.ReadAllText(openFileDialog.FileName);

}

private void Save\_Click(object sender, RoutedEventArgs e)//Обработчик кнопки сохранить

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

if (saveFileDialog.ShowDialog() == true)

File.WriteAllText(saveFileDialog.FileName, source.Text);

}

void solve(int timeOfCount)

{

List<DispTask> tasks; //лист всех процессов

try

{

tasks = FileManager.getFromString(source.Text); //получаем из поля для ввода

int i = 0; // счетчик тактов

while (tasks.Count != 0) //пока ещё есть невыполненные процессы

{

List<DispTask> loadTaskInTik = new List<DispTask>(); //список задач в готовности

if (i % timeOfCount == 0)

{

loadTaskInTik = tasks.Where(x => x.activateTime <= i).ToList(); //заполняем список

DispTask ActiveTask = loadTaskInTik.Find(y => y.time == loadTaskInTik.Max(x => x.time)); //выбираем активный процесс

loadTaskInTik.Remove(ActiveTask);

TaskBox taskBox = new TaskBox();

taskBox.Content = ActiveTask.name;

taskBox.Width = 55;

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

ActiveBar.Children.Add(taskBox); //отображаем

ActiveTask.work++; //такт проработал

//выводи список процессов в готовности

StackPanel stackPanel = new StackPanel();

stackPanel.Orientation = Orientation.Vertical;

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

if (loadTaskInTik.Count == 0)

{

taskBox = new TaskBox();

taskBox.Content = " ";

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

else

{

foreach (DispTask task in loadTaskInTik)

{

if (task.name != "null")

{

taskBox = new TaskBox();

taskBox.Content = task.name;

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

}

}

LoadBar.Children.Add(stackPanel);

//удаляем процесс, который отраотал своё

if (ActiveTask.work >= ActiveTask.time)

{

tasks.Remove(ActiveTask);

}

}

i++;

}

createTimeLine(i); //рисуем ось

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

Листинг 3. FileManager.cs

using System;

using System.Collections.Generic;

using System.Windows;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dispatcher

{

public static class FileManager

{

public static List<DispTask> getFromFile(string path)

{

List<DispTask> tasks = new List<DispTask>();

try

{

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

string str = sr.ReadLine();

var parameters = str.Split(' ');

tasks.Add(new DispTask(parameters[0], Convert.ToInt32(parameters[1]), Convert.ToInt32(parameters[2])));

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

return tasks;

}

public static List<DispTask> getFromString(string str)

{

List<DispTask> tasks = new List<DispTask>();

var strings = str.Split('\n');

foreach (string i in strings)

{

var param = i.Split(' ');

tasks.Add(new DispTask(param[0], Convert.ToInt32(param[1]), Convert.ToInt32(param[2])));

}

return tasks;

}

}

}

Листинг 4. DispTask.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dispatcher

{

public class DispTask

{

public string name;

public int time; //время исполнения

public int activateTime; //время активациии процесса

public int work; //сколько уже выполнялся

public DispTask(string Name, int Time, int ActivateTime)

{

name = Name;

time = Time;

activateTime = ActivateTime;

}

}

}

Листинг 5. TaskBox.xaml

<UserControl x:Class="Dispatcher.TaskBox"

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

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

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

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

xmlns:local="clr-namespace:Dispatcher"

mc:Ignorable="d"

d:DesignHeight="50" d:DesignWidth="100">

<Grid>

<Border Background="Transparent" BorderThickness="1" BorderBrush="Black">

<Grid>

<Label HorizontalAlignment="Center" VerticalAlignment="Center" Content="{Binding Title}"/>

</Grid>

</Border>

</Grid>

</UserControl>

Листинг 6. TaskBox.xaml.cs

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 Dispatcher

{

/// <summary>

/// Логика взаимодействия для TaskBox.xaml

/// </summary>

public partial class TaskBox : UserControl

{

public string Title{ get; set;}

public TaskBox()

{

InitializeComponent();

this.DataContext = this;

}

}

}

Пример работы программы

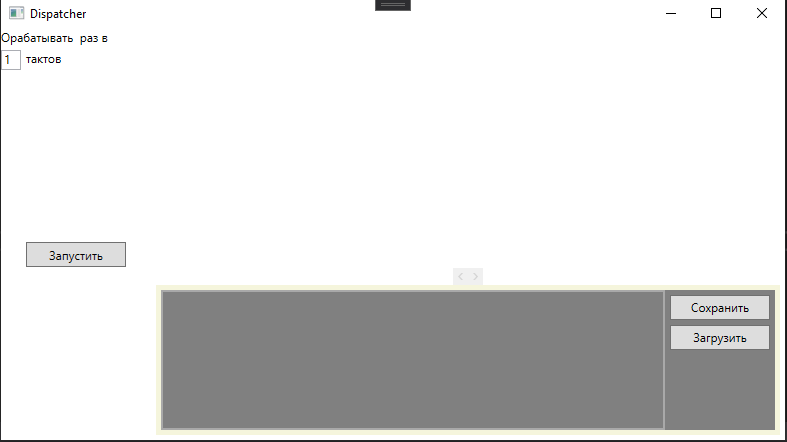


Рисунок 1. Начальное окно

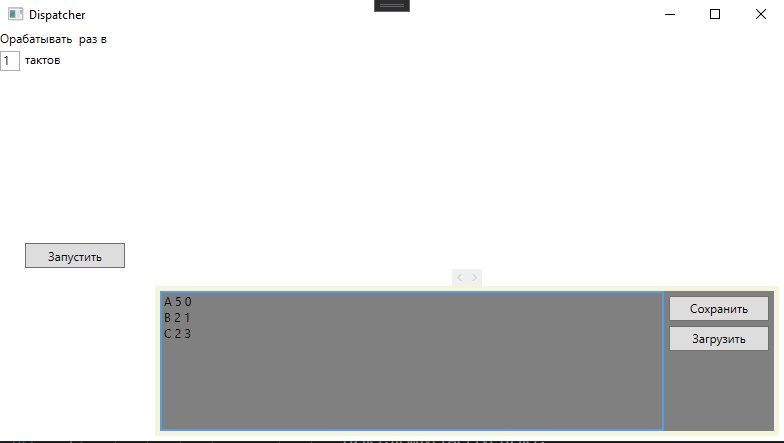


Рисунок 2. Загруженный файл с примером

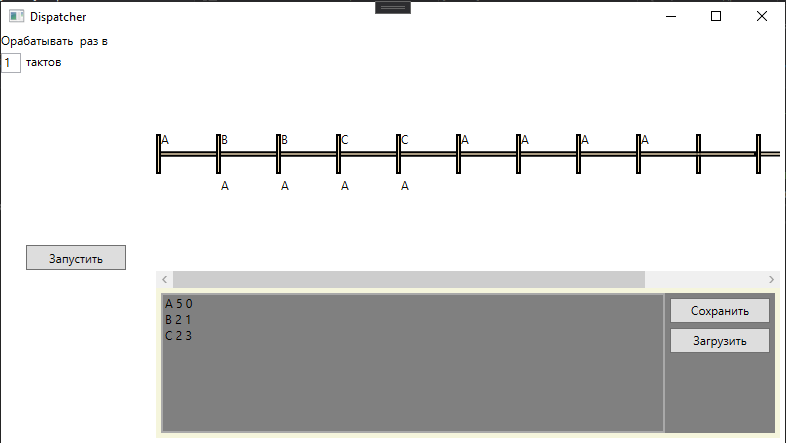


Рисунок 3. Выполненный расчёт