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 |  |  |  | Д.Р. Шитов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

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

**Вариант 13**

Диспетчер на основе дисциплины LCFS (last come – first served – последним пришёл - первым обслужился). Данная дисциплина является вытесняющей и характерна для обработки прерываний.

*Листинг 1*

*XAML разметка*

<Window x:Class="Dispatcher13.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:Dispatcher13"

mc:Ignorable="d"

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

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition Width="130"/>

</Grid.ColumnDefinitions>

<Border Margin="5"

Grid.Column="0"

VerticalAlignment="Center"

BorderBrush="Black"

BorderThickness="3"

Background="LavenderBlush">

<ScrollViewer 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 40 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>

<StackPanel Grid.Column="1"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<Button Content="Старт"

Margin="2"

Width="70"

Click="Start\_Click"/>

<Button Content="Выбрать"

Margin="2"

Width="70"

Click="Load\_Click"/>

</StackPanel>

</Grid>

</Window>

Листинг 2

Класс MainWindow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using Microsoft.Win32;

using System.IO;

namespace Dispatcher13

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

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

static string Path;

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

public MainWindow()

{

InitializeComponent();

}

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

{

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

Border VerticalLine = new Border { Background = Brushes.Pink, 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.Pink, 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 Calculate() // функция обработки

{

if (tasks.Count() !=0) //проверка на дурака

{

int i = 0; //делаем счётчик тактов

while (tasks.Count() != 0) // пока список не пустой

{

List<MyTask> loadTasks = tasks.Where(z => z.activeTime <= i).ToList(); // определяем список задач, которые готовы к выполнению

MyTask activeTask = loadTasks.Find(z => i - z.activeTime == loadTasks.Min(x => i - x.activeTime)); //из них выбираем активную задачу

loadTasks.Remove(activeTask); //убираем чтобы не выводить

activeTask.work++;//счётчик отработанных тактов увеличиваем

if (activeTask.work >= activeTask.time) // если всё отработал

{

tasks.Remove(activeTask);// то домой

}

TaskBox taskBox = new TaskBox(); //отображаем активную задачу

taskBox.Title = activeTask.name;

taskBox.Width = 55;

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

ActiveBar.Children.Add(taskBox);

//отображаем задачи в состоянии готовности

StackPanel stackPanel = new StackPanel();

stackPanel.Orientation = Orientation.Vertical;

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

if (loadTasks.Count == 0)

{

taskBox = new TaskBox();

taskBox.Content = " ";

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

else

{

foreach (MyTask task in loadTasks)

{

if (task.name != "null")

{

taskBox = new TaskBox();

taskBox.Content = task.name;

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

}

}

LoadBar.Children.Add(stackPanel);

i++;

}

createTimeLine(i); // генерация отображения оси

}

else

{

MessageBox.Show("Загрузите входной файл!");

}

}

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

{

LoadBar.Children.Clear();

ActiveBar.Children.Clear();

Calculate();

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

tasks.Clear();

try

{

var strs = File.ReadLines(openFileDialog.FileName);

foreach (string i in strs)

{

var param = i.Split(' ');

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

}

MessageBox.Show("Файл загружен");

}

catch

{

MessageBox.Show("Проблемы с вводом");

}

}

}

}

public class MyTask

{

public string name;

public int time; //время исполнения

public int activeTime; //время активациии процесса

public int work; //сколько уже выполнялся

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

{

name = Name;

time = Time;

activeTime = AciveTime;

}

}

}

**Скриншоты работы программы**



Рисунок 1 – Тестовые значения

(имя, время обработки, время выполнения)

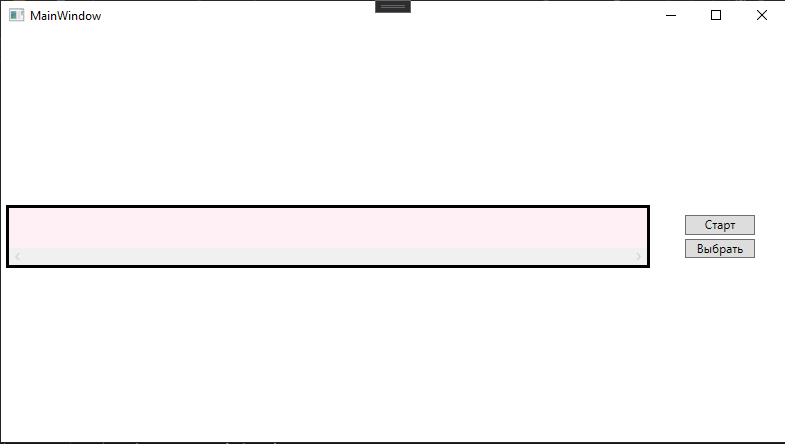


Рисунок 2 – начальное окно

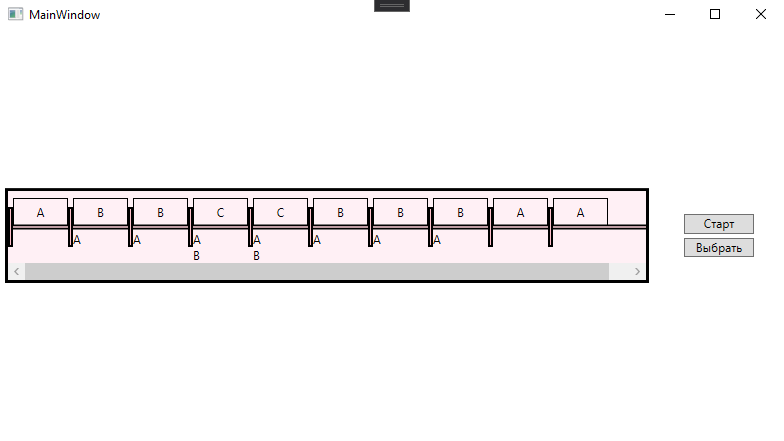


Рисунок 3 – результат расчёта