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 |  |  |  | Н.С. Горбунов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

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

**Вариант 7**

Диспетчер на основе дисциплины с динамическими приоритетами, изменяющимися по формуле Pri=Pr0i + ai\*exp(t). Дисциплины на основе динамических приоритетов задач. Для каждой задачи задаётся начальное значение приоритета, которое затем изменяется во времени. Таким образом, приоритет задачи есть функция времени. Конкретный вид таких функций может быть разный, но общая их направленность состоит в том, что, чем дольше задача находится в очереди готовности, тем выше становится её приоритет.

Это позволяет гарантировать обслуживание как коротких, так и длинных процессов. Изменение приоритетов во времени осуществляет диспетчер. Через определённый период времени (интервал пересчёта ) диспетчер пересчитывает значения приоритетов задач, которые в момент пересчёта находятся в состоянии готовности.

Затем диспетчером определяется готовая задача с максимальным приоритетом. Если её приоритет оказывается выше приоритета текущей выполняемой задачи, диспетчер снимает с выполнения текущую задачу и переводит её в очередь готовности и предоставляет процессор готовой новой задаче. Через следующий интервал пересчёта вышеуказанная процедура повторяется.

**Код программы:**

*Листинг 1*

*XAML разметка*

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

xmlns:MyNamespace="clr-namespace:ProcessModeling"

mc:Ignorable="d"

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

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="100"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Border Background="AliceBlue" BorderBrush="Beige" BorderThickness="5" AllowDrop="true" Drop="txtTarget\_Drop" />

<Grid>

<StackPanel >

<TextBlock Text="Перерасчёт приоритетов 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>

<Button Content="Запустить расчёт" Height="25" Width="100" Click="Button\_Click"/>

</StackPanel>

</Grid>

<ScrollViewer Margin="5" Grid.Row="1" 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="0 140 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>

</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;

using System.IO;

namespace ProcessModeling

{

/// <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 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);

}

}

void solve(int timeOfCount)

{

List<MyTask> tasks; //лист всех процессов

try

{

tasks = FileManager.getFromFile(Path); //получаем из файла

List<MyTask> ActiveTasks = new List<MyTask>(); // порядок работающих процессов

List<List<MyTask>> LoadTasks = new List<List<MyTask>>(); // список загруженных процессов

int i = 0; // счетчик тактов

while (tasks.Count != 0)

{

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

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

if (i % timeOfCount == 0)

{

foreach (MyTask task in tasks)

{

task.calculatePr(i); //рассчёт приоритетов

}

}

double maxPr = tasks[0].Pr;

string maxName = tasks[0].name;

foreach (MyTask task in tasks) //поиск максимального приоритета

{

if (task.Pr > maxPr)

{

maxPr = task.Pr;

maxName = task.name;

}

}

MyTask ActiveTask = tasks.Find(x => x.name == maxName);

ActiveTask.work++;

ActiveTasks.Add(ActiveTask);

foreach (MyTask task in tasks)

{

if (task.activeTime <= i && task.work < task.time && task.name != maxName) //определяем, какие процессы необходимо загрузить

{

loadTaskInTik.Add(task);

}

if (task.work >= task.time) //а какие уже отраотали своё

{

deletelist.Add(task);

}

}

if (loadTaskInTik.Count() == 0)

{

loadTaskInTik.Add(new MyTask("null", 0, 0, 0, 0));

}

LoadTasks.Add(loadTaskInTik);

foreach (MyTask task in deletelist)

{

tasks.Remove(task);

}

i++;

}

createTimeLine(ActiveTasks.Count);

foreach (MyTask task in ActiveTasks) //рисуем

{

TaskBox taskBox = new TaskBox();

taskBox.Content = task.name;

taskBox.Width = 55;

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

ActiveBar.Children.Add(taskBox);

}

foreach (List<MyTask> line in LoadTasks)

{

StackPanel stackPanel = new StackPanel();

stackPanel.Orientation = Orientation.Vertical;

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

foreach (MyTask task in line)

{

if (task.name != "null")

{

TaskBox taskBox = new TaskBox();

taskBox.Content = task.name;

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

}

LoadBar.Children.Add(stackPanel);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

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

Path = files[0];

}

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

{

LoadBar.Children.Clear();

ActiveBar.Children.Clear();

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

}

}

public class MyTask

{

public string name;

public int time; //время исполнения

public int activeTime; //время активации процесса

double a; // коэффициент приоритета

double Pr0; //изначальный приоритет

public double Pr; //текущий приоритет

public int work; //сколько уже выполнялся

public MyTask(string Name, int Time, int AciveTime, double A, double Pr0)

{

name = Name;

time = Time;

activeTime = AciveTime;

a = A;

this.Pr0 = Pr0;

Pr = Pr0;

}

public void calculatePr(int t)

{

Pr = Pr0 + a \* System.Math.Exp(t);

}

}

public static class FileManager

{

public static List<MyTask> getFromFile(string path)

{

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

try

{

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

string str = sr.ReadLine();

var parameters = str.Split(' ');

tasks.Add(new MyTask(parameters[0], Convert.ToInt32(parameters[1]), Convert.ToInt32(parameters[2]), Convert.ToDouble(parameters[3]), Convert.ToDouble(parameters[4])));

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

return tasks;

}

}

}

**Скриншоты работы программы**

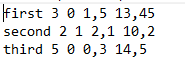


Рисунок 1 – Тестовые значения

(имя, время обработки, коэффициент а, начальный приоритет)

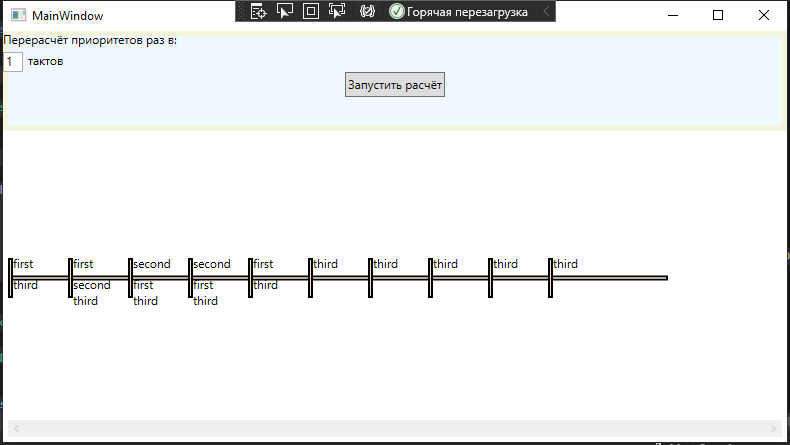


Рисунок 2 – результат расчёта

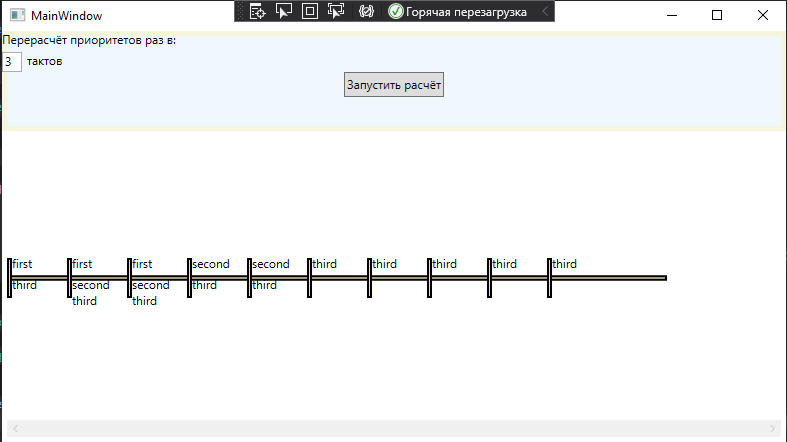


Рисунок 3 – результат расчёта