МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра СКС

Лабораторна робота №3

з дисципліни

системне програмне забезпечення

на тему:

**«Дослідження дисциплін обслуговування заявок при обмежених ресурсах»**

Виконав студент Перевірив:

Групи КВ-64 \_\_\_\_\_\_\_\_\_\_\_

### Подольський Сергій Валентинович

залікова книжка № КВ6415 \_\_\_\_\_\_\_\_\_\_\_\_\_(бали)

#### VІІ семестр

Київ-2009

## **1. Постановка задачі**

1. Згідно варіанту виконати по тактову візуалізацію системи обслуговування заявок з розподіленим ресурсом, параметрами системи є діапазон ваг заявок, діапазон інтервалів виникнення наступної заявки (інтенсивність вхідного потоку), діапазон пріоритетів (для пріоритетних дисциплін обслуговування). В систему може надходити будь-яка кількість заявок.
2. Якщо система обслуговування з чергами, то кількість черг не більша 32. Якщо система обслуговування з пріоритетами, то число пріоритетів не більш ніж 32.
3. Побудувати графіки залежності середнього часу очікування від інтенсивності вхідного потоку заявок та залежність проценту простою ресурсу від інтенсивності вхідного потоку заявок.
4. Для пріоритетних систем побудувати графік залежності середнього часу очікування від пріоритету при фіксованої інтенсивності вхідного потоку заявок. Для систем без пріоритетів побудувати графік дисперсії часу очікування при фіксованій інтенсивності вхідного потоку заявок.
5. Пояснити форму графіків.

**2. Опис роботи дисципліни обслуговування**

Згідно варіанту виконати по тактову візуалізацію системи обслуговування заявок з розподіленим ресурсом, параметрами системи є діапазон ваг заявок, діапазон інтервалів виникнення наступної заявки (інтенсивність вхідного потоку), діапазон пріоритетів (для пріоритетних дисциплін обслуговування). В систему може надходити будь-яка кількість заявок.

Якщо система обслуговування з чергами, то кількість черг не більша 32. Якщо система обслуговування з пріоритетами, то число пріоритетів не більш ніж 32.

Побудувати графіки залежності середнього часу очікування від інтенсивності вхідного потоку заявок та залежність проценту простою ресурсу від інтенсивності вхідного потоку заявок.

Для пріоритетних систем побудувати графік залежності середнього часу очікування від пріоритету при фіксованої інтенсивності вхідного потоку заявок. Для систем без пріоритетів побудувати графік дисперсії часу очікування при фіксованій інтенсивності вхідного потоку заявок.

Пояснити форму графіків.

**3. Лістинг основної частини програми**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace TaskScheduler

{

public partial class FormMain : Form

{

public FormMain()

{

InitializeComponent();

// Add comboBox value to user settings

comboBox.DataBindings.Add("SelectedIndex", TaskScheduler.Properties.Settings.Default, "SchedulingAlgorithm");

numericUpDownIntervalVariation.Maximum = numericUpDownMaxIncomingInterval.Value;

}

private void buttonProceed\_Click(object sender, EventArgs e)

{

// Clear all previous results

listBox.Items.Clear();

foreach (Series series in chart.Series)

series.Points.Clear();

List<DataPoint> points0 = new List<DataPoint>();

List<DataPoint> points1 = new List<DataPoint>();

// Start scheduler with different incoming task intensities

for (int incomingInterval = (int)numericUpDownMaxIncomingInterval.Value; incomingInterval >= 1; incomingInterval--)

{

// Choose scheduler algorithm

Scheduler scheduler;

if (comboBox.SelectedItem == null) return;

string algorithm = (string)comboBox.SelectedItem.ToString();

if (algorithm == "FIFO") scheduler = new SchedulerFIFO();

else if (algorithm == "RR") scheduler = new SchedulerRR();

else if (algorithm == "LLQ") scheduler = new SchedulerLLQ();

else return;

// Regard only completed task or all tasks that entered

List<Task> taskList = checkBoxRegardCompleted.Checked ? scheduler.TasksLeaved : scheduler.TasksEntered;

// Add event to list box

if (checkBoxMakeEventList.Checked && incomingInterval == (int)numericUpDownIntervalVariation.Value)

scheduler.TaskCompleted += new Scheduler.TaskCompletedEventHandler(delegate(Task task, int time)

{

listBox.Items.Add(time.ToString() + ":\tTask completed");

});

// FORCE SCHEDULING

Random random = new Random(DateTime.Now.Millisecond);

int next = random.Next(1, incomingInterval);

for (scheduler.Time = 0; scheduler.Time < numericUpDownTotalTime.Value; scheduler.Time++)

if (--next == 0)

{

Task task = new Task(random.Next(1, (int)numericUpDownMaxTaskWeight.Value), (int)numericUpDownPriorityCount.Value, (int)numericUpDownPriorityClasses.Value);

task.EndTime = (int)numericUpDownTotalTime.Value;

scheduler.AddTask(task);

next = random.Next(1, incomingInterval);

// Add event to list box

if (checkBoxMakeEventList.Checked && incomingInterval == (int)numericUpDownIntervalVariation.Value)

listBox.Items.Add(scheduler.Time.ToString() + "\tTask entered");

}

// Add points to chart

double intensity = (double)taskList.Count / (double)numericUpDownTotalTime.Value;

points0.Add(new DataPoint(intensity, scheduler.IdleTime));

points1.Add(new DataPoint(intensity, taskList.Average(new Func<Task, double>(delegate(Task task)

{

return task.IdleTime;

}))));

// Draw variation

if (incomingInterval == (int)numericUpDownIntervalVariation.Value)

{

int MaxTaskIdle = taskList.Max<Task>(new Func<Task, int>(delegate(Task task)

{

return task.IdleTime;

}));

chart.ChartAreas[1].AxisX.Maximum = MaxTaskIdle + (int)numericUpDownVariationStep.Value - MaxTaskIdle % (int)numericUpDownVariationStep.Value;

for (int idle = 0; idle <= MaxTaskIdle; idle += (int)numericUpDownVariationStep.Value)

chart.Series[2].Points.AddXY(idle + (int)numericUpDownVariationStep.Value / 2, taskList.Count(new Func<Task, bool>(delegate(Task task)

{

return task.IdleTime >= idle && task.IdleTime < idle + (int)numericUpDownVariationStep.Value;

})));

}

}

points0.Sort(new Comparison<DataPoint>(delegate(DataPoint p1, DataPoint p2)

{

return p1.XValue >= p2.XValue ? 1 : -1;

}));

points1.Sort(new Comparison<DataPoint>(delegate(DataPoint p1, DataPoint p2)

{

return p1.XValue >= p2.XValue ? 1 : -1;

}));

foreach (DataPoint point in points0)

chart.Series[0].Points.AddXY(point.XValue, point.YValues[0]);

foreach (DataPoint point in points1)

chart.Series[1].Points.AddXY(point.XValue, point.YValues[0]);

}

/// <summary>

/// Check constraints

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void numericUpDownMaxIncomingInterval\_ValueChanged(object sender, EventArgs e)

{

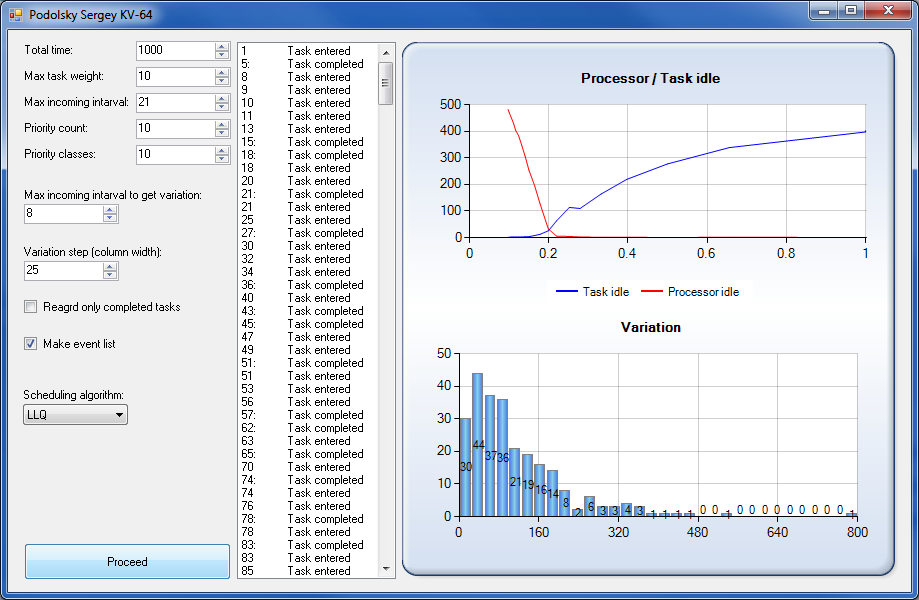
numericUpDownIntervalVariation.Maximum = numericUpDownMaxIncomingInterval.Value;

}

}

}

**4. Результати роботи програми**



**3. Переваги та недоліки досліджуваної дисципліни обслуговування**

Перевагою даної дисципліни обслуговування є те, що вона враховує класи пріоритетів та пріоритети заявок, що поступають до черги, не перериваючи при цьому виконання поточної заявки.

Недолік даної дисципліни полягає в тому, що вона ігнорує заявки з найнижчим класом пріоритету та найнижчим пріоритетом. Таким чином, при досить високій інтенсивності вхідних заявок вона може взагалі не потрапити на обслуговування.