**Национальный Технический Университет Украины**

**«Киевский Политехнический Институт»**

**Факультет информатики и вычислительной техники**

**Кафедра вычислительной техники**

**Лабораторная работа №3**

**«Исследование дисциплин обслуживания заявок при ограниченных ресурсах»**

Выполнил: студент ІІІ курса

группы ИВ-92

Петрук В.

Киев – 2012

## Задание на работу

1. Согласно варианту выполнить потактовую визуализацию системы обслуживания заявок с разделяемым ресурсом, параметрами системы являются диапазон весов заявок, диапазон интервалов появления очередной заявки (интенсивность входного потока), диапазон приоритетов (для приоритетных дисциплин обслуживания). В систему может поступать любое количество заявок.
2. Если система обслуживания с очередями, то количество очередей не более 32. Если система обслуживания с приоритетами, то число приоритетов не более 32.
3. Построить графики зависимости среднего времени ожидания от интенсивности входного потока заявок и зависимость процента простоя ресурса от интенсивности входного потока заявок.
4. Для приоритетных систем построить график зависимости среднего времени ожидания от приоритета при фиксированной интенсивности входного потока заявок. Для систем без приоритетов построить график дисперсии времён ожидания при фиксированной интенсивности входного потока заявок.
5. Объяснить форму графиков.

## Варианты

Вариант выбирается по номеру зачётной книжки:

последние\_две\_цифры mod 16 + 1

CBWFQ

## Отчёт

1. Описание работы дисциплины обслуживания.

***CBWFQ ( Class-Based Weighted Fair Queuing)***

Алгоритм побудови черг, що базуються на класах, називається CBWFQ (Class-Based Weighted Fair Queuing). Алгоритм CBWFQ надає механізм управління при перевантаженнях. Параметри, які характеризують клас, ті ж, що і у разі WFQ (тільки замість TоS використовується пріоритет). На відміну від WFQ тут можна в широких межах перерозподіляти полосу пропускання між потоками. Для виділення класу можуть притягуватися ACL (Access Control List) або навіть номер вхідного інтерфейсу. Кожному класу ставиться у відповідність черга.Даний алгоритм гарантує смугу лише в умовах перевантаження. Всього може бути визначено 64 класи. Нерозподілена полоса може використовуватися потоками згідно їх пріоритетам.

1. Листинг программы.

using System;

using System.Collections;

using System.Collections.Generic;

namespace laba3\_spo

{

public class FifoContainer

{

public static Timer1 timerCounter = new Timer1(0);

private int \_allTime = 0;

private int \_counter = 0;

private int \_numOfFifo = 0;

private FifoNode[] \_dbase = null;

private Timer1 \_timer = new Timer1(0);

private Chanel \_chanel = new Chanel();

public FifoContainer(int sizeOfFifo)

{

\_numOfFifo = sizeOfFifo;

\_dbase = new FifoNode[sizeOfFifo];

displayFifoHeader();

}

public Boolean addFIFO(int sizeFifo, int countOfPackage, int prior)

{

if (\_counter < \_numOfFifo)

{

\_dbase[\_counter] = new FifoNode(sizeFifo, countOfPackage, prior);

\_counter++;

return true;

} else {

ContainerForm.errorList.addError(new ErrorNode("Eror: number of FIFO queue which you want no fit your input data!"+\_numOfFifo.ToString()));

return false;

}

}

public void addFIFO(FifoNode fn)

{

if (\_counter <= \_numOfFifo)

{

\_dbase[\_counter] = fn;

\_counter++;

}

else

{

ContainerForm.errorList.addError(new ErrorNode("Eror: number of FIFO queue which you want no fit your input data!"));

}

}

private void displayFifoHeader()

{

ContainerForm.form.dataGridView1.ColumnCount = \_numOfFifo\*4 + 1;

ContainerForm.form.dataGridView1.Columns[0].Name = "№";

ContainerForm.form.dataGridView1.ColumnHeadersHeight = 10;

for (int i = 1; i <= \_numOfFifo; i++ )

{

ContainerForm.form.dataGridView1.Columns[i\*4-3].Name = "Num";

ContainerForm.form.dataGridView1.Columns[i\*4-2].Name = "S";

ContainerForm.form.dataGridView1.Columns[i\*4-1].Name = "F";

ContainerForm.form.dataGridView1.Columns[i\*4].Name = "ClassName";

}

}

public void loadDataToGrid()

{

int max = 0;

for (int i = 0; i < \_numOfFifo; i++ )

{

if(\_dbase[i].getEnd() > max)

{

max = \_dbase[i].getEnd();

}

}

for(int j = 0 ; j < max; j++ )

{

var row = new string[\_numOfFifo \* 4+1];

row[0] = (j + 1).ToString();

for(int i = 1; i <= \_numOfFifo; i++)

{

try

{

Package pack = \_dbase[i - 1].getNext();

row[i \* 4 - 3] = pack.numberP.ToString();

row[i \* 4 - 2] = pack.timeStart.ToString();

row[i \* 4 - 1] = pack.timeEnd.ToString();

row[i \* 4 ] = pack.layer.getClassName();

} catch(Exception ex)

{

row[i \* 4 - 3] = "-";

row[i \* 4 - 2] = "-";

row[i \* 4 - 1] = "-";

row[i \* 4] = "-";

}

}

ContainerForm.form.dataGridView1.Rows.Add(row);

}

for (int i = 0; i < \_numOfFifo; i++)

{

\_dbase[i].setStartPos(0);

}

}

public void SortAll()

{

for (int i = 0; i < \_numOfFifo; i++)

{

\_dbase[i].SortByClass();

}

for (int i = 0; i < \_numOfFifo; i++)

{

\_dbase[i].setStartPos(0);

}

loadDataToGrid();

}

public void loadFIFOToGrid()

{

ContainerForm.form.dataGridView2.ColumnCount = 4;

ContainerForm.form.dataGridView2.Columns[0].Name = "№";

ContainerForm.form.dataGridView2.Columns[1].Name = "Priority";

ContainerForm.form.dataGridView2.Columns[2].Name = "Size Of FIFO";

ContainerForm.form.dataGridView2.Columns[3].Name = "Num of Package";

for (int i = 0; i < \_numOfFifo; i++)

{

var row = new string[4];

row[0] = i.ToString();

row[1] = \_dbase[i].getPriority().ToString();

row[2] = \_dbase[i].sizeofarr.ToString();

row[3] = \_dbase[i].getEnd().ToString();

ContainerForm.form.dataGridView2.Rows.Add(row);

}

for (int i = 0; i < \_numOfFifo; i++)

{

\_dbase[i].setStartPos(0);

}

}

public int getNumOfFIFO()

{

return \_numOfFifo;

}

public void startTransmit()

{

var tempFN = new Package[\_numOfFifo];

for(int i = 0; i < \_numOfFifo; i++)

{

try

{

if (\_dbase != null)

{

tempFN[i] = \_dbase[i].getNext();

if(tempFN[i].getTimeToProcessing() >= \_timer.getTimeNode())

{

}else

{

\_chanel.addToChanel(tempFN[i]);

}

}

}catch

{

}

}

}

public void Run()

{

int max = 0;

for (int i = 0; i < \_numOfFifo; i++)

{

if (\_dbase[i].getEnd() > max)

{

max = \_dbase[i].getEnd();

}

}

SortFIFO();

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

{

var list = new Package[\_numOfFifo];

int k = 0;

for (int j = 0; j < \_numOfFifo; j++)

{

try

{

list[k] = \_dbase[j].getNext();

if(list[k] != null)

k++;

}catch(Exception ex)

{

}

}

SortList(list, k);

for (int j = 0; j < k; j++)

{

\_chanel.addToChanel(list[j]);

}

}

ContainerForm.form.dataGridView3.ColumnCount = 3;

ContainerForm.form.dataGridView3.Columns[0].Name = "№";

ContainerForm.form.dataGridView3.Columns[1].Name = "ClassName";

ContainerForm.form.dataGridView3.Columns[2].Name = "Time";

List < Package > chanList = \_chanel.getList();

Package[] plist = chanList.ToArray();

List<int[]> listres = new List<int[]>();

for (int i = 0; i < chanList.Count; i++)

{

var row = new string[3];

row[0] = i.ToString();

row[1] = plist[i].layer.getClassName();

var rowInt = new int[4]{DateTime.Now.Hour - plist[i].time.hour,

DateTime.Now.Minute - plist[i].time.minute,

DateTime.Now.Second - plist[i].time.second,

DateTime.Now.Millisecond - plist[i].time.milisec};

listres.Add(rowInt);

row[2] = (Math.Abs(rowInt[0])) + " hour - " +

(Math.Abs(rowInt[1])) + " minute - " +

(Math.Abs(rowInt[2])) + " sec - " +

(Math.Abs(rowInt[3])) + " milisec";

ContainerForm.form.dataGridView3.Rows.Add(row);

}

//\_chanel.Intensivnost(listres);

}

public void RunTime()

{

for (int i = 0; i < \_numOfFifo; i++)

{

for (int j = 0; j < \_dbase[i].getEnd(); j++)

{

Package pack = \_dbase[i].getNext();

pack.time = new Time();

pack.time.start();

}

}

for (int i = 0; i < \_numOfFifo; i++)

{

\_dbase[i].setStartPos(0);

}

}

private Package[] SortList(Package[] \_list, int n)

{

int max = \_list[0].layer.getPriority();

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

{

for (int j = 0; j < n; j++)

{

if (max <= \_list[j].layer.getPriority())

{

Package temp = \_list[j];

\_list[j] = \_list[i];

\_list[i] = temp;

}

max = \_list[i].layer.getPriority();

}

}

return \_list;

}

private void SortFIFO()

{

int max = \_dbase[0].getPriority();

for (int i = 0; i < \_numOfFifo; i++)

{

for (int j = 0; j < \_numOfFifo; j++)

{

if (max <= \_dbase[j].getPriority())

{

FifoNode temp = \_dbase[j];

\_dbase[j] = \_dbase[i];

\_dbase[i] = temp;

}

max = \_dbase[i].getPriority();

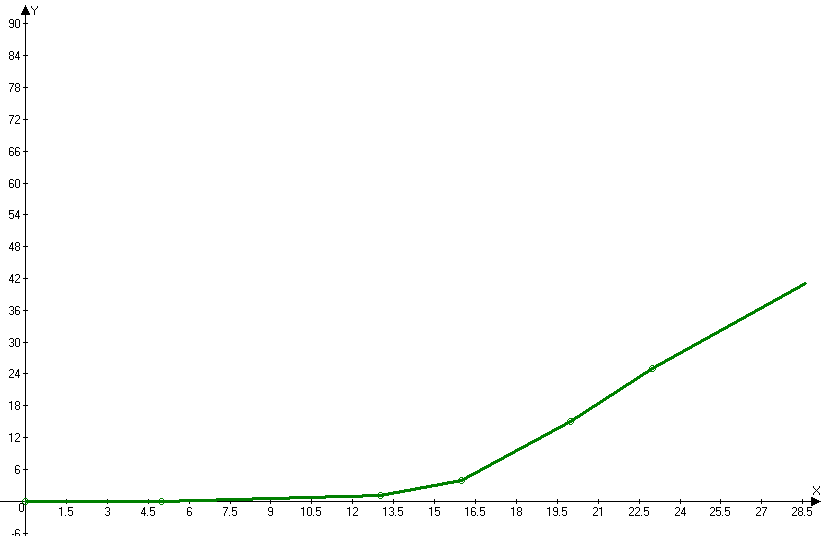
}

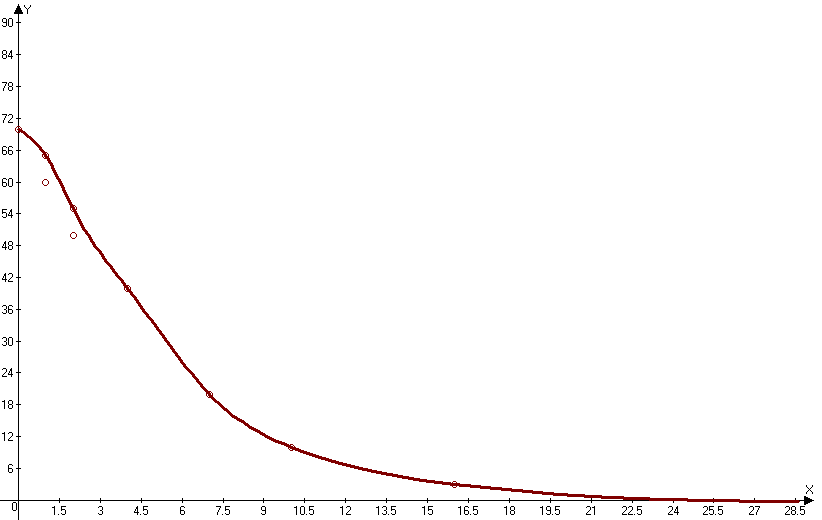
}

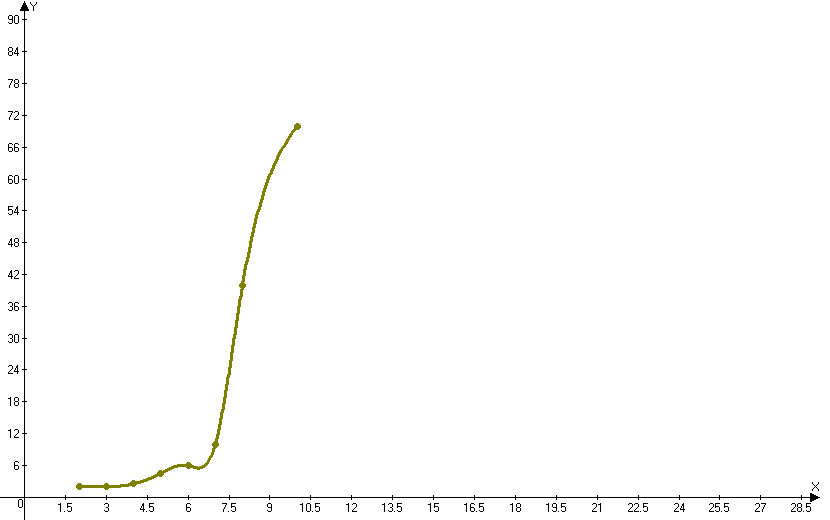
}

}

}

1. ГрафикиИнтенсивность ко времени ожидания

Простой ресурс от интенсивности



Среднее ковариация от времени ожидания заявки

1. Достоинства и недостатки исследуемой дисциплины обслуживания.

К достоинствам можно отнести эффективное распределение времени для каждой заявки так, как каждая заявка имеет свой приоритет, следовательно при правильных приоритетах выполнение заявок будет достаточно эффективно.

Недостатки не обнаружены. Используется в ЦИСКА системах.