Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра математического и аппаратного обеспечения информационных систем

Лабораторная работа № 2

по дисциплине “Операционные системы”

на тему:

**Генерация и загрузка задач ПО «Модель ОС»**

Выполнил:

студент группы ИВТ-11-17

Садовников А.В.

Принял:

Ковалев С.В.

Чебоксары, 2020

**Цель работы:** Разработать подпрограммы генерации и загрузки задач главной формы ПО «Модель ОС».

**Задание:**

1. Описать параметры процессора и дать им значение по умолчанию.

2. Описать параметры системы и дать им значение по умолчанию.

3. Описать тип для состояния процесса («Отсутствует», «Готов», «Выполняется» и т.д.).

4. Описать слово состояния процесса (включает идентификатор задания, размер задания, счетчик команд, состояние процесса и т.д.).

5. Задать массив слов состояний процессов (РSW).

6. Описать для нового задания переменную РSW\_Task типа «Слово состояния процесса».

7. Инициализировать массив слов состояний процессов.

8. Реализовать подпрограмму «Генерация нового задания» со значениями параметров по умолчанию.

9. Реализовать подпрограмму «Проверка наличия доступной памяти».

10. Реализовать подпрограмму «Загрузка нового задания».

11. Реализовать цикл загрузки заданий для начала моделирования.

12. Изменить индикацию для вывода слов состояний процессов.

**Выполнение**:

1. Значения по умолчанию хранятся в файле config/StartOS:

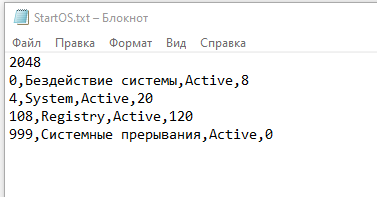


Рис 1 – настройки ОС

«2048» - размер памяти ОС

«0» - PID задачи

«Бездействие системы» - Название задачи

«Active» - Состояние задачи после запуска ОС

«8» - Требуемая операционная память для выполнения задачи

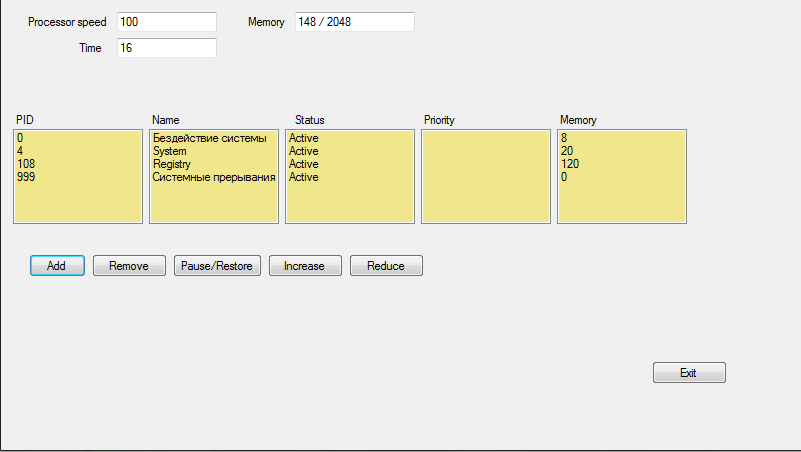


Рис 2 – Состояние ОС после инициализации

1. Подпрограмма «Генерация нового задания»:

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

{

Process process = new Process();

string newName;

int newPID;

int newMemory;

FormAddNewProcess addForm = new FormAddNewProcess();

if(addForm.ShowDialog() == DialogResult.OK)

{

newName = addForm.getNewName();

newPID = addForm.getNewPID();

newMemory = addForm.getNewMemory();

process.set(newPID, newName, newMemory);

listBoxPID.Items.Add(newPID);

listBoxName.Items.Add(newName);

listBoxStatus.Items.Add("Active");

//priority

listBoxMemory.Items.Add(newMemory);

statusMemory();

}

}

1. Подпрограмма «Проверка наличия доступной памяти»:

public void statusMemory()

{

int currentMemory = 0;

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

{

if(listBoxStatus.Items[i].ToString() == "Active")

{

currentMemory += Convert.ToInt32(listBoxMemory.Items[i]);

}

}

if(currentMemory > allMemory)

{

MessageBox.Show("Кончилась память");

}

textBoxAboutMemory.Text = currentMemory + " / " + allMemory;

}

1. Подпрограмма «Загрузка системных заданий»

public void StartSettings()

{

string path = "config/StartOS.txt";

try

{

StreamReader sr = new StreamReader(path);

{

allMemory = Convert.ToInt32(sr.ReadLine());

string line;

while((line = sr.ReadLine()) != null)

{

string[] parameters = line.Split(',');

listBoxPID.Items.Add(Convert.ToInt32(parameters[0]));

listBoxName.Items.Add(parameters[1]);

listBoxStatus.Items.Add(parameters[2]);

listBoxMemory.Items.Add(Convert.ToInt32(parameters[3]));

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

statusMemory();

}

Количество доступной памяти до добавления нового процесса(рис 3):

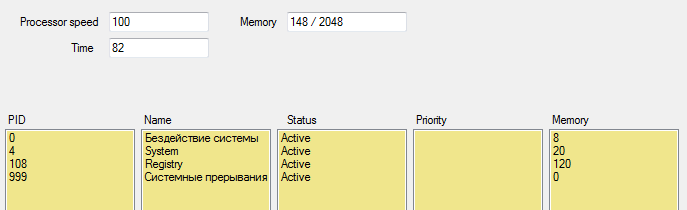


Рис 3 – до добавления

Количество доступной памяти после добавления нового процесса с названием «Visual Studio»(рис 4):

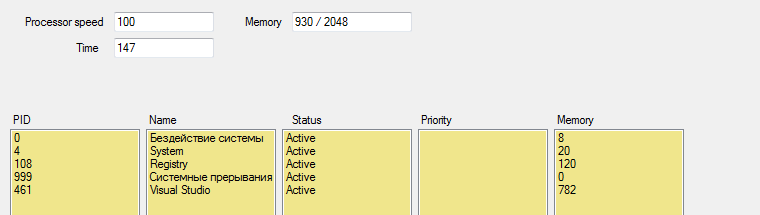


Рис 4 – после добавления