МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра СКС

Лабораторна робота №3

з дисципліни

операційні системи

на тему:

**«Керування пам’яттю»**

Виконав студент Перевірив:

Групи КВ-64 \_\_\_\_\_\_ Дроб’язко І.П.

### Подольський Сергій Валентинович

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

#### V семестр

Київ-2008

1. **Постановка задачі.**

Розробити модель алгоритму управління пам’яттю **з фіксованими розділами** (без використання зовнішньої пам’яті).

Кількість розділів менша, ніж кількість процесів. Процеси створюють загальну чергу до розділів пам’яті. Структуризація адресного простору – **лінійна**. Розміри процесів задаються випадково.

Продемонструвати процес перетворення заданої віртуальної адреси у фізичну.

1. **Опис принципу роботи програми та її використання.**

Програма реалізована в середовищі розробки Microsoft Visual Studio 2008 на платформі DOT NET 3.5 мовою програмування C#.

Програма працює наступним чином: спочатку формується модель фізичної пам’яті, яка складається з заданих користувачем розділів заданого розміру.

Щоб додати новий процес, користувач задає лише розмір пам’яті, який йому необхідний. Нумерація розділів і процесів ведеться автоматично, починаючи з нуля.

При додаванні нового процесу іде пошук розділу, в який може бути завантажений процес. Пошук здійснюється за наступними критеріями: по порядку аналізується кожний розділ на наявність в ньому завантаженого іншого процесу. Якщо розділ «вільний» і його розмір не менший, ніж розмір процесу, пошук припиняється і процес завантажується в даний розділ. Якщо розділу, який задовольняє зазначеним вимогам не знайдено, процес стає до загальної черги, яка відображається в програмі у вигляді таблиці з інформацією щодо процесів, які перебувають в загальній черзі.

Список розділів в програмі реалізовано в двох режимах:

1. табличному;
2. графічному.

Табличний режим являє собою таблицю, в якій міститься інформація щодо кожного розділу і процесів, які завантажені в розділи. Користувач може виділити будь-який розділ в таблиці і звільнити його від процесу, натиснувши відповідну кнопку, тобто завершити процес, який завантажений в даний розділ. Після завершення відповідного процесу відбувається спроба завантаження процесів з черги у пріоритеті їх розташування в загальній черзі, тобто таблиці. Якщо завантаження процесу в вивільнений розділ можливе, то одразу після завершення процесу виділеного користувачем розділу в нього завантажується процес з загальної черги.

Графічний режим являє собою графічне зображення фізичної пам’яті, розділеної на розділи у вигляді прямокутників. Якщо в розділ завантажений процес, то область занятої ним пам’яті у області розділу зображується інакшим кольором. При виділенні деякого розділу в таблиці відповідна йому графічна область в графічному зображенні пам’яті також виділяється пунктиром.

При натисненні лівої кнопки миші в деяку точку на графічному зображенні пам’яті з’являється спливаюче повідомлення з поточною віртуальною та фізичною адресо, на яку був навадений курсор, а також процес перетворення віртуальної адреси у фізичну. У повідомленні міститься інформація про відповідний розділ та процес, в межах якого знаходиться дана поточна адреса. Якщо курсор був наведений у область пам’яті розділу, не зайняту процесом, то в повідомлення лише зазначається, що задана адреса пам’яті даного розділу вільна.

1. **Текст програми.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace MemoryMapper

{

public partial class FormMemoryMapper : Form

{

const string none = "<нема>";

int size = 0;

public FormMemoryMapper()

{

InitializeComponent();

numericUpDownSectionNumber.Maximum = numericUpDownSectionOffset.Maximum = numericUpDownSectionSize.Maximum = numericUpDownProcessNumber.Maximum = numericUpDownProcessSize.Maximum = decimal.MaxValue;

buttonAddSection\_Click(null, null);

}

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

{

dataGridViewSections.Rows.Add(numericUpDownSectionNumber.Value++, numericUpDownSectionOffset.Value, numericUpDownSectionSize.Value, none, 0);

textBoxMemory.Text = (numericUpDownSectionOffset.Value = size += Convert.ToInt32(numericUpDownSectionSize.Value)).ToString();

Add();

}

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

{

dataGridViewQueue.Rows.Add(numericUpDownProcessNumber.Value++, numericUpDownProcessSize.Value);

Add();

}

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

{

DataGridViewCellCollection cells = dataGridViewSections.SelectedRows[0].Cells;

cells[ColumnProcess.Name].Value = none;

cells[ColumnMemoryUsed.Name].Value = 0;

Add();

dataGridViewSections\_SelectionChanged(null, null);

}

private void dataGridViewSections\_SelectionChanged(object sender, EventArgs e)

{

buttonClear.Enabled = (dataGridViewSections.SelectedRows.Count > 0) && (Convert.ToInt32(dataGridViewSections.SelectedRows[0].Cells[ColumnMemoryUsed.Name].Value) != 0);

area.Invalidate();

}

void Add()

{

foreach (DataGridViewRow process in dataGridViewQueue.Rows)

foreach (DataGridViewRow section in dataGridViewSections.Rows)

if ((Convert.ToInt32(section.Cells[ColumnMemoryUsed.Name].Value) == 0) && (Convert.ToInt32(section.Cells[ColumnSectionSize.Name].Value) >= Convert.ToInt32(process.Cells[ColumnProcessSize.Name].Value)))

{

section.Cells[ColumnProcess.Name].Value = process.Cells[ColumnProcessNumber.Name].Value;

section.Cells[ColumnMemoryUsed.Name].Value = process.Cells[ColumnProcessSize.Name].Value;

dataGridViewQueue.Rows.Remove(process);

dataGridViewSections\_SelectionChanged(null, null);

area.Invalidate();

return;

}

area.Invalidate();

}

private void labelArea\_Paint(object sender, PaintEventArgs e)

{

double scale = (double)area.Height / (double)size;

for (int section = 0; section < dataGridViewSections.Rows.Count; section++)

{

int sectionOffset = Convert.ToInt32(dataGridViewSections.Rows[section].Cells[ColumnOffset.Name].Value);

int sectionSize = Convert.ToInt32(dataGridViewSections.Rows[section].Cells[ColumnSectionSize.Name].Value);

int memoryUsed = Convert.ToInt32(dataGridViewSections.Rows[section].Cells[ColumnMemoryUsed.Name].Value);

Rectangle sectionRectangle = new Rectangle(0, (int)(scale \* sectionOffset), area.Width, (int)(scale \* sectionSize));

Rectangle processRectangle = new Rectangle(0, (int)(scale \* sectionOffset), area.Width, (int)(scale \* memoryUsed));

e.Graphics.FillRectangle(Brushes.White, sectionRectangle);

e.Graphics.FillRectangle(Brushes.Aquamarine, processRectangle);

e.Graphics.DrawRectangle(new Pen(Brushes.Black, 2), sectionRectangle);

if (section == dataGridViewSections.SelectedRows[0].Index)

e.Graphics.FillRectangle(new System.Drawing.Drawing2D.HatchBrush(HatchStyle.DottedDiamond, Color.Gray, Color.Transparent), sectionRectangle);

if (memoryUsed > 0)

{

StringFormat stringFormat = new StringFormat();

stringFormat.Alignment = StringAlignment.Center;

e.Graphics.DrawString("Процес " + dataGridViewSections.Rows[section].Cells[ColumnProcess.Name].Value.ToString(), new Font("Times New Roman", 10), Brushes.Black, processRectangle, stringFormat);

}

if (sectionRectangle.Contains(mouse))

{

string s = "Розділ " + section.ToString() + "\n" + "Адреса розілу: " + sectionOffset.ToString() + "\n\n";

if (processRectangle.Contains(mouse))

{

int physicalOffset = (int)((double)mouse.Y / scale);

int virtualOffset = physicalOffset - sectionOffset;

s += "Процес " + dataGridViewSections.Rows[section].Cells[ColumnProcess.Name].Value.ToString() + "\n";

s += "Віртуальна адреса: " + virtualOffset.ToString() + "\n";

s += "Фізична адреса: " + virtualOffset.ToString() + "+" + sectionOffset.ToString() + "=" + physicalOffset.ToString();

}

else s += "Вільна область пам'яті";

toolTip.Show(s, area, mouse);

Pen pen = new Pen(Color.Red, 2);

pen.DashStyle = DashStyle.Dash;

e.Graphics.DrawLine(pen, 0, mouse.Y, Width, mouse.Y);

}

}

}

Point mouse = new Point(-1, -1);

private void area\_MouseDown(object sender, MouseEventArgs e)

{

mouse = e.Location;

area.Invalidate();

}

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

{

Application.Restart();

}

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

{

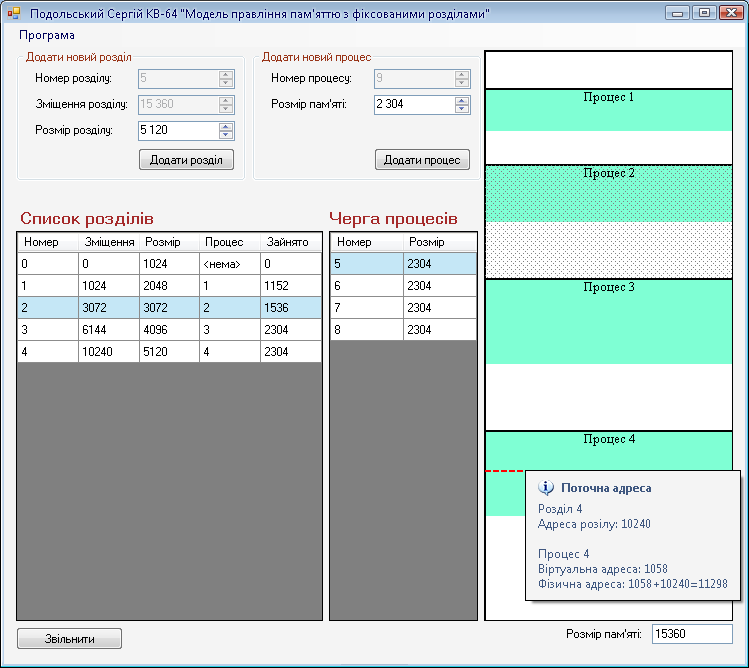
Close();

}

}

}

1. **Результати роботи програми:**

****

Використана література

1. MSDN 2008
2. Карлі Ватсон – С#.NET