Министерство образования и науки РФ

Тверской государственный технический университет

Кафедра электронных вычислительных машин

Отчет к лабораторной работе № 3

Выполнил: Морозов А.Н.

Группа Б.ИВТ.ПРОМ.23.01

Принял: Рачишкин А. А.

Тверь 2024

**Цель работы:**

Разобрать и дополнить программное приложения для просмотра и контроля процессов в системе.

**Задачи:**

1. Проанализировать программный код.

2. Спроектировать общую архитектуру приложения.

3. Определиться с интерфейсом приложения.

a. Улучшить его в сравнении с примером.

b. Добавить возможность сохранения в лог файл текущих процессов.

4. Реализовать результаты 3-х задач в программном коде.

**Доп. задачи:**

5. Реализовать отображения других характеристик процесса.

6. Проанализировать результат и выявить алгоритмическую ошибку.

a. Исправить ошибку.

7. Создать дополнительный функционал на ваш взгляд подходящей программе.

**Код программы:**

ListViewItemComparer.cs

using System;

using System.Collections;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

class ListViewItemComparer : IComparer

{

private int columnIndex;

public int ColumnIndex

{

get

{

return columnIndex;

}

set

{

columnIndex = value;

}

}

private SortOrder sortDirection;

public SortOrder SortDirection

{

get

{

return sortDirection;

}

set

{

sortDirection = value;

}

}

public ListViewItemComparer()

{

sortDirection = SortOrder.None;

}

public int Compare(object x, object y)

{

ListViewItem listViewItemX = x as ListViewItem;

ListViewItem listViewItemY = y as ListViewItem;

int result;

switch(columnIndex)

{

case 0:

result = string.Compare(listViewItemX.SubItems[columnIndex].Text, listViewItemY.SubItems[columnIndex].Text, false);

break;

case 1:

double valueX = double.Parse(listViewItemX.SubItems[columnIndex].Text);

double valueY = double.Parse(listViewItemY.SubItems[columnIndex].Text);

result = valueX.CompareTo(valueY);

break;

default:

result = string.Compare(listViewItemX.SubItems[columnIndex].Text, listViewItemY.SubItems[columnIndex].Text, false);

break;

}

if(sortDirection == SortOrder.Descending)

{

return -result;

}

else

{

return result;

}

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Diagnostics;

using System.Management;

using Microsoft.VisualBasic;

namespace WindowsFormsApp1

{

public partial class ProcessManager: Form

{

private List<Process> processes = null;

private ListViewItemComparer comparer = null;

public ProcessManager()

{

InitializeComponent();

}

private void GetProcesses()

{

processes.Clear();

processes = Process.GetProcesses().ToList<Process>();

}

private void RefreshProcessesList()

{

DisplayField.Items.Clear();

double memSize = 0;

foreach (Process process in processes)

{

PerformanceCounter pc = new PerformanceCounter();

pc.CategoryName = "Process";

pc.CounterName = "Working Set - Private";

pc.InstanceName = process.ProcessName;

memSize = (double)pc.NextValue() / (1000 \* 1000);

string[] row = new string[] {process.ProcessName.ToString(), Math.Round(memSize, 1).ToString()};

DisplayField.Items.Add(new ListViewItem(row));

pc.Close();

pc.Dispose();

}

Text = "Запущено процессов: " + processes.Count.ToString();

}

private void RefreshProcessesList(List<Process> processes, string keyword)

{

try

{

DisplayField.Items.Clear();

double memSize = 0;

foreach (Process process in processes)

{

if (process != null)

{

memSize = 0;

PerformanceCounter pc = new PerformanceCounter();

pc.CategoryName = "Process";

pc.CounterName = "Working Set - Private";

pc.InstanceName = process.ProcessName;

memSize = (double)pc.NextValue() / (1000 \* 1000);

string[] row = new string[] { process.ProcessName.ToString(), Math.Round(memSize, 1).ToString() };

DisplayField.Items.Add(new ListViewItem(row));

pc.Close();

pc.Dispose();

}

}

Text = $"Запущено процессов '{keyword}'" + processes.Count.ToString();

}

catch (Exception) { }

}

private void KillProcess(Process process)

{

process.Kill();

process.WaitForExit();

}

private void KillProcessAndChildren(int pid)

{

if(pid == 0)

{

return;

}

ManagementObjectSearcher searcher = new ManagementObjectSearcher(

"Select \* From Win32\_Process Where ParentProcessID=" + pid);

ManagementObjectCollection results = searcher.Get();

foreach (ManagementObject obj in results)

{

KillProcessAndChildren(Convert.ToInt32(obj["ProcessID"]));

}

try

{

Process p = Process.GetProcessById(pid);

p.Kill();

p.WaitForExit();

}

catch (ArgumentException) { }

}

private int GetParentProcessID(Process process)

{

int pid = 0;

try

{

ManagementObject managementObject = new ManagementObject("win32\_process.Handle='" + process.Id + "'");

managementObject.Get();

pid = Convert.ToInt32(managementObject["ParentProcessId"]);

}

catch (Exception) { }

return pid;

}

private void Form1\_Load(object sender, EventArgs e)

{

processes = new List<Process>();

GetProcesses();

RefreshProcessesList();

comparer = new ListViewItemComparer();

comparer.ColumnIndex = 0;

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

GetProcesses();

RefreshProcessesList();

}

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

{

try

{

if (DisplayField.SelectedItems[0] != null)

{

Process process = null;

string selectedItemText = DisplayField.SelectedItems[0].SubItems[0].Text;

foreach (Process proc in processes)

{

if (proc.ProcessName == selectedItemText)

{

process = proc;

break;

}

}

KillProcess(process);

GetProcesses();

RefreshProcessesList();

}

}

catch(Exception) { }

}

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

{

try

{

if (DisplayField.SelectedItems[0] != null)

{

Process process = null;

string selectedItemText = DisplayField.SelectedItems[0].SubItems[0].Text;

foreach (Process proc in processes)

{

if (proc.ProcessName == selectedItemText)

{

process = proc;

break;

}

}

KillProcessAndChildren(GetParentProcessID(process));

GetProcesses();

RefreshProcessesList();

}

}

catch (Exception) { }

}

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

{

try

{

if (DisplayField.SelectedItems[0] != null)

{

Process process = null;

string selectedItemText = DisplayField.SelectedItems[0].SubItems[0].Text;

foreach (Process proc in processes)

{

if (proc.ProcessName == selectedItemText)

{

process = proc;

break;

}

}

KillProcessAndChildren(GetParentProcessID(process));

GetProcesses();

RefreshProcessesList();

}

}

catch (Exception) { }

}

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

{

string path = Interaction.InputBox("Введите имя программы", "Запуск новой задачи");

try

{

Process.Start(path);

}

catch (Exception) { }

}

private void InputField \_TextChanged(object sender, EventArgs e)

{

GetProcesses();

List<Process> filteredProcesses = new List<Process>();

foreach (Process process in processes)

{

if (process.ProcessName.ToLower().Contains(InputField.Text.ToLower()))

{

filteredProcesses.Add(process);

}

}

RefreshProcessesList(filteredProcesses, InputField.Text);

}

private void DisplayField \_ColumnClick(object sender, ColumnClickEventArgs e)

{

comparer.ColumnIndex = e.Column;

comparer.SortDirection = comparer.SortDirection == SortOrder.Ascending ? SortOrder.Descending : SortOrder.Ascending;

DisplayField.ListViewItemSorter = comparer;

DisplayField.Sort();

}

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

{

Application.Exit();

}

}

}

**Результат выполнения программы:**

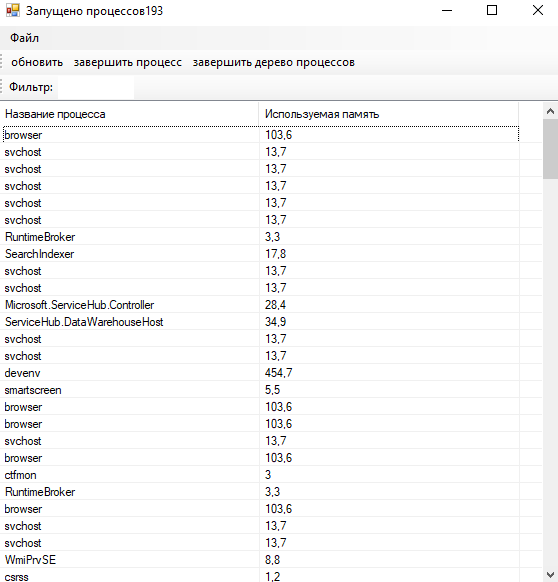


Рис 1. Результат выполнения программы

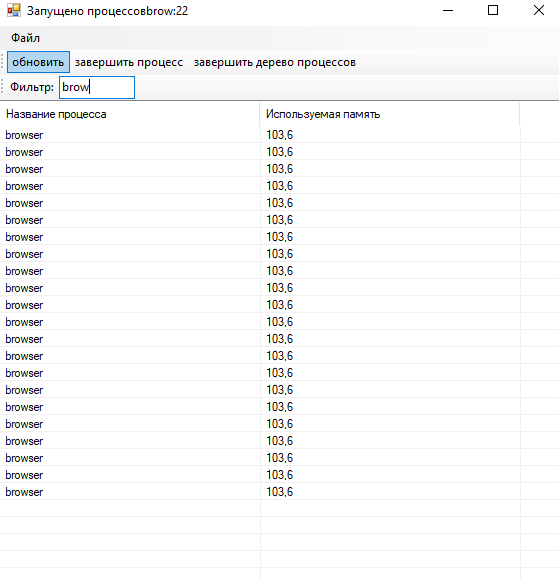


Рис 2. Результат работы фильтра