Phần 1: Giới thiệu Windows API, cách sử dụng các hàm Windows API trong ngôn ngữ lập trình C#

I/ Giới thiệu Windows API

1. API là gì?

Một giao diện lập trình ứng dụng (tiếng Anh là Application Program Interface, viết tắt API) là một [giao diện](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) mà một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu dịch vụ có thể được tạo ra từ các chương trình máy tính khác, và/hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng. Chẳng hạn, một chương trình máy tính có thể (và thường là phải) dùng các hàm API của [hệ điều hành](https://vi.wikipedia.org/wiki/H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) để xin cấp phát bộ nhớ và truy xuất tập tin. Nhiều loại hệ thống và ứng dụng thực hiện API, như các hệ thống đồ họa, cơ sở dữ liệu, mạng, dịch vụ web, và ngay cả một số trò chơi máy tính. Đây là phần mềm hệ thống cung cấp đầy đủ các chức năng và các tài nguyên mà các lập trình viên có thể rút ra từ đó để tạo nên các tính năng giao tiếp người- máy như: các trình đơn kéo xuống, tên lệnh, hộp hội thoại, lệnh bàn phím và các cửa sổ. Một trình ứng dụng có thể sử dụng nó để yêu cầu và thi hành các dịch vụ cấp thấp do hệ điều hành của máy tính thực hiện. Hệ giao tiếp lập trình ứng dụng giúp ích rất nhiều cho người sử dụng vì nó cho phép tiết kiệm được nhiều thời gian tìm hiểu các chương trình mới, do đó khích lệ mọi người dùng nhiều ứng dụng hơn.

Có ba chính sách đối với việc phát hành các API là:

* Cá nhân: Các API chỉ sử dụng nội bộ trong công ty.
* Đối tác: Chỉ các đối tác kinh doanh được sử dụng API.
* Công khai: Các API có sẵn cho mọi người sử dụng.

Một ví dụ điển hình của API công khai đó là Windows API. Microsoft luôn phân phối Windows API đi kèm theo hệ điều hành Microsoft Windows cho phép mọi người có thể sử dụng chúng.

1. Giới thiệu Windows API

Windows API là tập hợp các API lõi của Microsoft, có sẵn trong các hệ điều hành Microsoft Windows. Hầu hết các chương trình chạy trên Windows tương tác với Windows API. Windows API đã luôn phơi bày một phần lớn cấu trúc tầng dưới của hệ thống Windows cho các lập trình viên. Lợi ích của việc này chính là cung cấp cho các lập trình viên sự linh hoạt và sức mạnh trong lập trình ứng dụng.

Các hàm được cung cấp bởi Windows API có thể chia thành tám nhóm:

* Base Services: Cung cấp quyền truy cập các tài nguyên cơ bản của một hệ thống Windows. Gồm những thứ như hệ thống file, các thiết bị, tiến trình, luồng và xử lý lỗi. Các hàm này nằm trong kernel32.dll và KernelBase.dll.
* Advanced Services: Cung cấp quyền truy cập các hàm sâu vào nhân hệ thống. Gồm những thứ như Windows registry, tắt nguồn/khởi động lại hệ thống (hoặc huỷ), bắt đầu/dừng/tạo một Windows service, quản lý tài khoản người dùng. Các hàm này nằm trong advapi32.dll và advapires32.dll.
* Graphics Device Interface: Cung cấp các hàm xuất nội dung đồ hoạ ra màn hình, máy in và các thiết bị ra khác. Chúng nằm trong gdi32.dll.
* User Interface: Cung cấp các hàm tạo và quản lý màn hình windows và các điều khiển cơ bản. Chúng nằm trong user32.dll.
* Common Dialog Box Library: Cấp quyền truy cập đến một vài điều khiển cấp cao được cấp bởi hệ điều hành, nằm trong comctl32.dll.
* Windows Shell: Thành phần của Windows API cho phép ứng dụng truy cập vào các hàm cung cấp bởi shell hệ điều hành, các thành phần này nằm trong shell32.dll.
* Network Services: Cấp quyền truy cập các tuỳ chọn mạng khác nhau của hệ điều hành, nằm trong netapi32.dll.

II/ Cách sử dụng hàm Windows API trong ngôn ngữ lập trình C#

Để sử dụng Windows API trong C#, ta cần sử dụng namespace:

using System.Runtime.InteropServices;

Namespace này cho phép lập trình viên sử dụng thuộc tính DLL Import để sử dụng các API chưa được kiểm soát trong chương trình.

Sau đó, ta sẽ import file DLL vào chương trình và khai báo hàm theo cú pháp:

[DllImport(“Dll\_file\_name”)]

[public|private] static extern Return\_type Func\_name([type parameter,…]);

Trong đó:

* Dll\_file\_name: tên file dll muốn import.
* [public|private]: tuỳ chọn thuộc tính public (toàn cục) hoặc private (cục bộ).
* Return\_type: kiểu trả về.
* Func\_name: tên hàm.
* Type: kiểu dữ liệu của tham số.
* Parameter: tên tham số.
* [type parameter,…]: danh sách tham số của hàm.

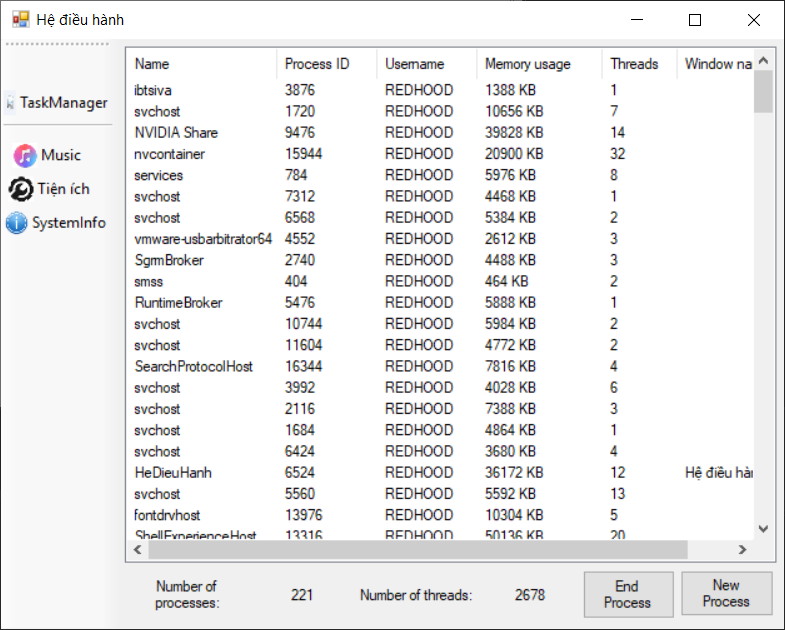
Phần 2: Viết ứng dụng sử dụng các hàm trong Windows API

**Đề tài:** Viết chương trình quản lý thông tin về hệ thống, bao gồm:

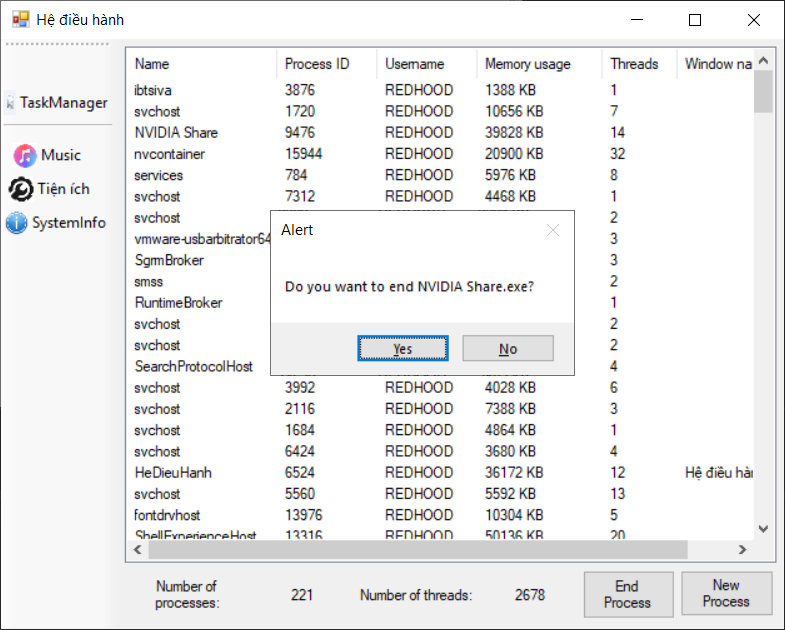
* Quản lý tiến trình (tương tự Task Manager). Xem các tiến trình đang chạy, có thể thêm mới tiến trình hoặc huỷ một tiến trình đang chạy.
* Các tiện ích: xem tình trạng kết nối mạng của thiết bị, xem tình trạng pin hiện tại của thiết bị, điều khiển âm lượng của thiết bị, thay đổi hình nền của thiết bị.
* Xem thông tin CPU.

I/ Quản lý tiến trình

1. **Hình ảnh minh hoạ**



Giao diện chính.



Chọn tiến trình và ấn End Process, hiện cửa sổ xác nhận.

Để tạo 1 tiến trình mới, ấn vào New Process và chọn đường dẫn tới tiến trình.

1. **Mã nguồn**

* Class TaskManager:

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace HeDieuHanh.Class

{

class TaskManager

{

public static TaskManager shared;

public static TaskManager getInstant()

{

if (shared == null)

{

shared = new TaskManager();

}

return shared;

}

//Trả về mảng các tiến trình

public Process[] getListProcess()

{

Process[] listProcess = Process.GetProcesses();

return listProcess;

}

//Khởi tạo ListView

public ListView setupListView(ListView processlist)

{

processlist.Columns.Add("Name");

processlist.Columns.Add("Process ID");

processlist.Columns.Add("Username");

processlist.Columns.Add("Memory usage");

processlist.Columns.Add("Threads");

processlist.Columns.Add("Window name");

processlist.Columns[0].Width = 120;

processlist.Columns[1].Width = 80;

processlist.Columns[2].Width = 80;

processlist.Columns[3].Width = 100;

processlist.Columns[4].Width = 60;

processlist.Columns[5].Width = 100;

return processlist;

}

}

}

* ucTaskManager:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using HeDieuHanh.Class;

using System.Diagnostics;

using System.Runtime.InteropServices;

namespace HeDieuHanh.GUI

{

public partial class ucTaskManager : UserControl

{

//Huỷ process

[DllImport("kernel32.dll", CharSet = CharSet.Unicode, SetLastError = true)]

static extern bool TerminateProcess(IntPtr hProcess, uint uExitCode);

//Lấy mã thoát của process

[DllImport("kernel32.dll", CharSet = CharSet.Unicode, SetLastError = true)]

static extern bool GetExitCodeProcess(IntPtr hProcess, out uint lpExitCode);

//Lấy ID của process

[DllImport("user32.dll", SetLastError = true)]

static extern uint GetWindowThreadProcessId(IntPtr hWnd, out uint lpdwProcessId);

//Lấy handle của process theo tên cửa sổ chương trình

[DllImport("user32.dll", SetLastError = true)]

static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

//Mở process, chỉnh sửa thuộc tính

[DllImport("kernel32.dll", SetLastError = true)]

public static extern IntPtr OpenProcess(ProcessAccessFlags processAccess, bool bInheritHandle, int processId);

//Đóng handle

[DllImport("kernel32.dll", SetLastError = true)]

static extern bool CloseHandle(IntPtr hObject);

TaskManager taskManager;

private static int processoldcount;

private static int threadoldcount = 0;

private static int columnclickcount = 0;

private static string selectedProcessName;

private static string selectedProcessWdnName;

private static IntPtr selectedProcessHandle;

//Tạo enumerate chứa hành động cho phép thực hiện với process

public enum ProcessAccessFlags : uint

{

All = 0x001F0FFF,

Terminate = 0x00000001,

CreateThread = 0x00000002,

VirtualMemoryOperation = 0x00000008,

VirtualMemoryRead = 0x00000010,

VirtualMemoryWrite = 0x00000020,

DuplicateHandle = 0x00000040,

CreateProcess = 0x000000080,

SetQuota = 0x00000100,

SetInformation = 0x00000200,

QueryInformation = 0x00000400,

QueryLimitedInformation = 0x00001000,

Synchronize = 0x00100000

}

public ucTaskManager()

{

InitializeComponent();

ListView lvProcess = new ListView();

taskManager = new TaskManager();

}

//Xoá sạch và tải lại ListView

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

{

lvProcess.Items.Clear();

setupUC();

}

//Hiển thị ListView

private void setupUC()

{

taskManager.setupListView(lvProcess);

showProcessList();

}

//Hiện danh sách tiến trình

private void showProcessList()

{

Process[] processList = taskManager.getListProcess();

processoldcount = processList.Count();

labelProcessCount.Text = processoldcount.ToString();

threadoldcount = 0;

lvProcess.BeginUpdate();

//Thêm các item mới chứa thông tin của process vào ListView

foreach (Process instance in processList)

{

string[] itemArray = new string[6];

ListViewItem newitem;

itemArray[0] = instance.ProcessName;

itemArray[1] = instance.Id.ToString();

itemArray[2] = Environment.MachineName;

itemArray[3] = ((instance.WorkingSet64 / 1024.0).ToString() + " KB");

itemArray[4] = instance.Threads.Count.ToString();

itemArray[5] = instance.MainWindowTitle.ToString();

newitem = new ListViewItem(itemArray);

lvProcess.Items.Add(newitem);

threadoldcount += Int32.Parse(itemArray[4]);

}

lvProcess.EndUpdate();

labelThreadCount.Text = threadoldcount.ToString();

}

//Xử lý sự kiện click chuột vào cột của ListView, sắp xếp theo tên

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

{

if (columnclickcount == 0)

{

lvProcess.Sorting = SortOrder.Ascending;

columnclickcount = 1;

}

else

{

lvProcess.Sorting = SortOrder.Descending;

columnclickcount = 0;

}

}

//Chọn item của ListView

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

{

selectedProcessName = lvProcess.SelectedItems[0].SubItems[0].Text;

selectedProcessWdnName = lvProcess.SelectedItems[0].SubItems[5].Text;

selectedProcessHandle = FindWindow(null, selectedProcessWdnName);

}

//Chạy chương trình mới

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

{

OpenFileDialog newProcessDialog = new OpenFileDialog();

newProcessDialog.Filter = "All File|\*.\*";

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

{

try

{

Process.Start(newProcessDialog.FileName);

}

catch (Exception exc)

{

MessageBox.Show(exc.Message, "Alert");

}

}

}

//Dừng chương trình

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

{

uint terminatedID;

//Lấy process id

GetWindowThreadProcessId(selectedProcessHandle, out terminatedID);

//Cấp quyền terminate process cho handle

selectedProcessHandle = OpenProcess(ProcessAccessFlags.Terminate, false, (int)terminatedID);

string messages = "Do you want to end " + selectedProcessName + ".exe?";

string caption = "Alert";

MessageBoxButtons buttons = MessageBoxButtons.YesNo;

DialogResult result;

result = MessageBox.Show(messages, caption, buttons);

if (result == DialogResult.Yes)

{

if (selectedProcessWdnName == "")

{

MessageBox.Show("Can not terminate system process", "Alert", MessageBoxButtons.OK);

}

else

{

uint exitcode;

//Lấy exit code của process

GetExitCodeProcess(selectedProcessHandle, out exitcode);

//Huỷ process

TerminateProcess(selectedProcessHandle, exitcode);

//Đóng handle

CloseHandle(selectedProcessHandle);

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

Process[] newList = taskManager.getListProcess();

int newcount = newList.Count();

//Refresh ListView nếu số lượng process thay đổi

if (newcount != processoldcount)

{

lvProcess.Items.Clear();

showProcessList();

}

int newthreadcount = 0;

//Đếm số lượng thread mới

foreach (Process instance in newList)

{

newthreadcount += instance.Threads.Count;

}

labelThreadCount.Text = newthreadcount.ToString();

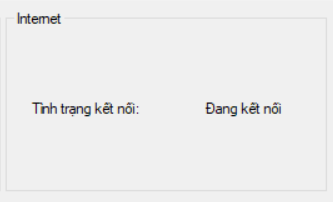
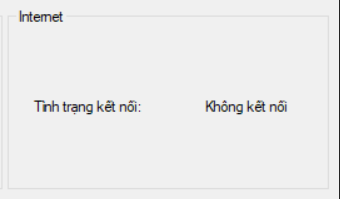
}

}

}

II/ Các tiện ích

1. Kiểm tra kết nối mạng
2. **Hình ảnh minh hoạ**

* Nếu đang có kết nối mạng, hiển thị “Đang kết nối”.
* Nếu không có kết nối mạng, hiển thị “Không kết nối”.

1. **Mã nguồn**

* Class Network:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace HeDieuHanh.Class

{

class Network

{

[DllImport("wininet.dll")]

private extern static bool InternetGetConnectedState(out int Description, int ReservedValue);

//hàm này sẽ trả về true nếu có kết nối internet và false nếu không

public static Network shared;

public static Network getInstant()

{

if (shared == null)

{

shared = new Network();

}

return shared;

}

public string getStateInternet()

{

int des;

if ((InternetGetConnectedState(out des, 0) == true))

{

return "Đang kết nối";

}

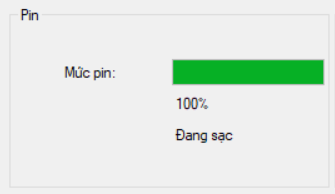
return "Không kết nối";

}

}

}

1. Kiểm tra tình trạng pin
2. **Hình ảnh minh hoạ**

****

* Khi không cắm sạc, hiển thị “Dùng pin”.
* Khi cắm sạc, hiển thị “Đang sạc”.

1. **Mã nguồn**

* Class Battery:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace HeDieuHanh.Class

{

class Battery

{

[DllImport("user32.dll", EntryPoint = "MessageBox")]

public static extern int MessageBox(int hwnd, string lpText, string lpCaption, int wType); //Hiển thị hộp thoại

//Lấy thông tin pin

PowerStatus powerStatus = SystemInformation.PowerStatus;

public static Battery shared;

public static Battery getInstant()

{

if (shared == null)

{

shared = new Battery();

}

return shared;

}

public int getBatteryLifePercent()

{

int capacityBattery = (int)(powerStatus.BatteryLifePercent \* 100);

return capacityBattery;

}

public string getBatteryStatus()

{

string batteryStatus = "";

switch (powerStatus.PowerLineStatus)

{

case PowerLineStatus.Offline:

batteryStatus = "Dùng pin";

break;

case PowerLineStatus.Online:

batteryStatus = "Đang sạc";

break;

default:

batteryStatus = "Tình trạng không rõ";

break;

}

return batteryStatus;

}

public string getPercentBattery()

{

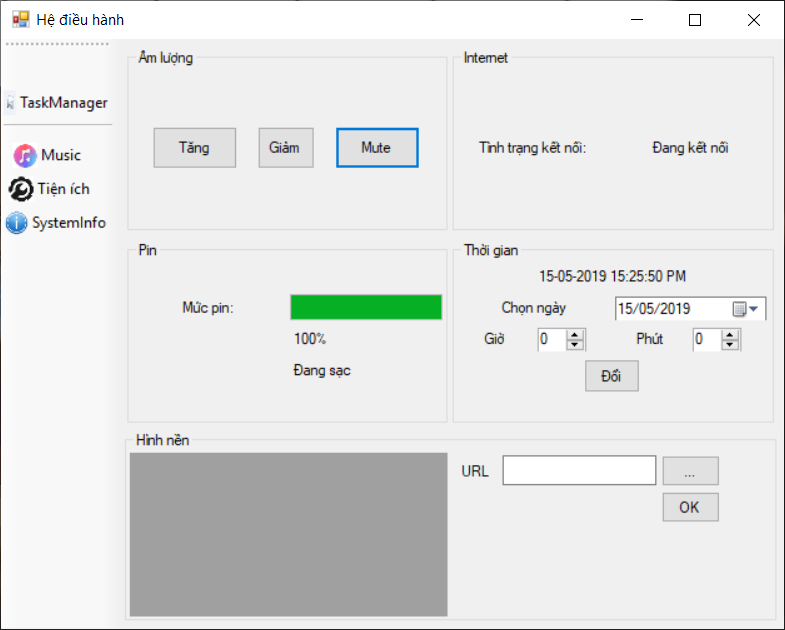
return ((int)(powerStatus.BatteryLifePercent \* 100)).ToString() + "%";

}

}

}

1. Thay đổi âm lượng
2. **Hình ảnh minh hoạ**

****

1. **Mã nguồn**

* Class Volume:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace HeDieuHanh.Class

{

class Volume

{

// Khai báo các tham số hệ thống

private const int VK\_VOLUME\_MUTE = 0xAD;

private const int VK\_VOLUME\_DOWN = 0xAE;

private const int VK\_VOLUME\_UP = 0xAF;

[DllImport("user32.dll")]

private static extern void keybd\_event(byte bVk, byte bScan, uint dwFlags, int dwExtraInfo);

public static Volume shared;

public static Volume getInstant()

{

if (shared == null)

{

shared = new Volume();

}

return shared;

}

//Tăng âm lượng

public void volumeUp()

{

keybd\_event((byte)VK\_VOLUME\_UP, 0, 0, 0);

}

//Giảm âm lượng

public void volumeDown()

{

keybd\_event((byte)VK\_VOLUME\_DOWN, 0, 0, 0);

}

//Tắt âm

public void mute()

{

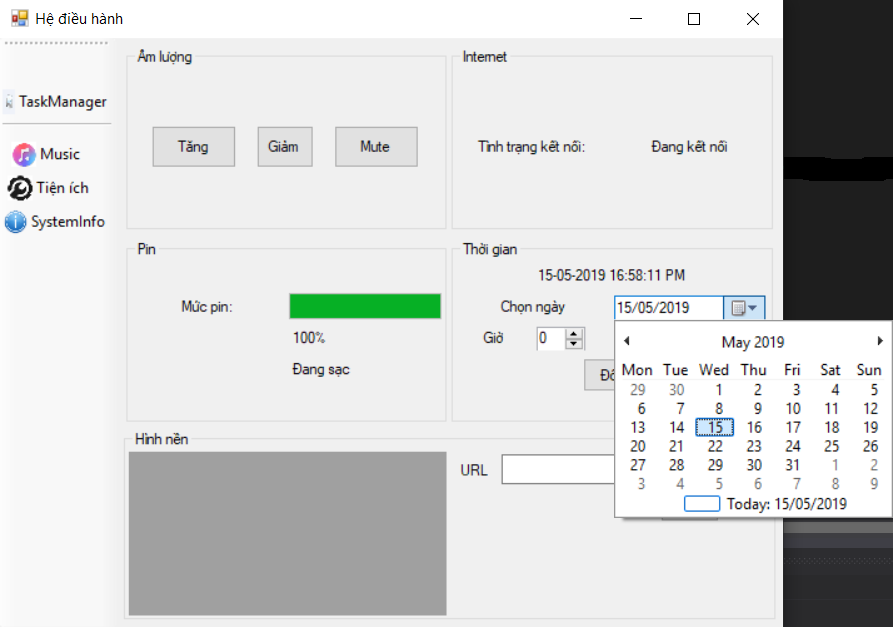
keybd\_event((byte)VK\_VOLUME\_MUTE, 0, 0, 0);

}

}

}

1. Thay đổi ngày giờ thiết bị
2. **Hình ảnh minh hoạ**

****

1. **Mã nguồn**

* Class Datetime:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace HeDieuHanh.Class

{

class Datetime

{

[DllImport("Kernel32.dll", SetLastError = true)]// import Kernel32.dll;

public extern static uint SetSystemTime(ref SYSTEMTIME lpSystemTime);// khai báo hàm api sử dụng;

public struct SYSTEMTIME // định nghĩa struct SYSTEMTIME để truyền vào hàm api SetSystemTime;

{

public short wYear;// biến lưu năm, có giá trị hợp lệ trong khoảng 1601-30827;

public short wMonth;// biến lưu tháng, có giá trị hợp lệ trong khoảng 1-12;

public short wDayOfWeek;// biến lưu ngày trong tuần, có giá trị hợp lệ trong khoảng từ 0-6 tương đương CN-thứ bảy;

public short wDay;// biến lưu ngày trong tháng, có giá trị hợp lệ trong khoảng 1-31;

public short wHour;// biến lưu giờ, có giá trị hợp lệ trong khoảng 0-23;

public short wMinute;// biến lưu phút, có giá trị hợp lệ trong khoảng 0-59;

public short wSecond;// biến lưu giây, có gía trị hợp lệ trong khoảng 0-59;

public short wMilliseconds;// biến lưu mili giây, có giá trị hợp lệ trong khoảng 0-999;

}

public static Datetime shared;

public static Datetime getInstant()

{

if (shared == null)

{

shared = new Datetime();

}

return shared;

}

public string getTime() // hàm lấy thời gian hiện tại của hệ thống, trả về 1 string;

{

return DateTime.Now.ToString("dd-MM-yyyy HH:mm:ss tt");

}

public void setTime(short year, short month, short day, short hour, short minute)// hàm đặt thời gian hệ thống; yêu cầu đầu vào theo thứ tự năm, tháng, ngày, giờ, phút

{

SYSTEMTIME st = new SYSTEMTIME();// tạo biến kiểu SYSTEMTIME để lưu các giá trị;

st.wYear = year;

st.wMonth = month;

st.wDayOfWeek = 0;

st.wDay = day;

st.wHour = hour;

st.wMinute = minute;

st.wSecond = 0;

st.wMilliseconds = 0;

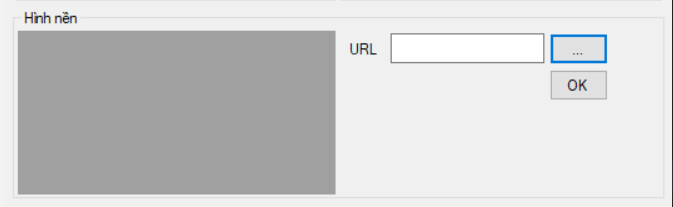
SetSystemTime(ref st);// truyền các giá trị vào để thay đổi thời gian hệ thống;

}

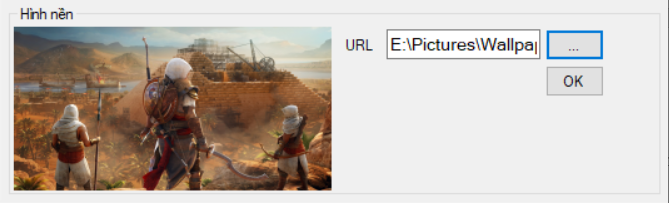
}

}

1. Thay đổi hình nền
2. **Hình ảnh minh hoạ**

****

* Sau khi chọn được hình nền, hiển thị hình minh hoạ ở khung dưới:



1. **Mã nguồn**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Runtime.InteropServices;

namespace HeDieuHanh.Class

{

class Wallpaper

{

//khai báo thư viện của hệ thống

[DllImport("user32.dll", CharSet = CharSet.Auto)]

private static extern int SystemParametersInfo(int uAction, int uParam, string lpvParam, int fuWinini);

private const int SPI\_SETDESKWALLPAPER = 20; // cài đặt hình ảnh màn hình

private const int SPIF\_SENDWININICHANGE = 0x1; //căng ảnh toàn màn hình

private const int SPIF\_UPDATEINIFILE = 0x2; // cập nhật hình ảnh

//Hàm & biến lấy đường dẫn của ảnh

public Wallpaper(string url)

{

\_url = url;

}

private string \_url;

public string Url

{

get { return \_url; }

set { \_url = value; }

}

//đổi hình nền bằng đường dẫn ảnh

public void ChangeBackground()

{

SystemParametersInfo(SPI\_SETDESKWALLPAPER, 0, \_url, SPIF\_SENDWININICHANGE);

}

}

}

1. Mã nguồn chung của tiện ích

* ucUltilities:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using HeDieuHanh.Class;

namespace HeDieuHanh.GUI

{

public partial class ucUtilities : UserControl

{

static int volumeValue;

public ucUtilities()

{

InitializeComponent();

}

public bool \_runThread = true;

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

{

setupForm();

}

//Tạo form

void setupForm()

{

nmrHour.Maximum = 24;

nmrMinute.Maximum = 59;

prgBattery.Maximum = 100;

prgBattery.Minimum = 0;

prgBattery.Value = Battery.getInstant().getBatteryLifePercent();

lblStateBattery.Text = Battery.getInstant().getBatteryStatus();

lblCapacityBattery.Text = Battery.getInstant().getPercentBattery();

lblStateNetwork.Text = Network.getInstant().getStateInternet();

lblDatetime.Text = Datetime.getInstant().getTime();

}

private void timer\_Tick(object sender, EventArgs e)

{

setupForm();

}

//Tăng âm

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

{

Volume.getInstant().volumeUp();

}

//Giảm âm

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

{

Volume.getInstant().volumeDown();

}

//Tắt âm

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

{

Volume.getInstant().mute();

}

//Đổi ngày giờ

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

{

short year = short.Parse(dtp.Value.Date.Year.ToString());

short month = short.Parse(dtp.Value.Date.Month.ToString());

short day = short.Parse(dtp.Value.Date.Day.ToString());

short hour = short.Parse(nmrHour.Value.ToString());

short minute = short.Parse(nmrMinute.Value.ToString());

try

{

Datetime.getInstant().setTime(year, month, day, hour, minute);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

//Chọn ảnh

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

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Image File|\*jpg;\*gif;\*png";

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

{

txtUrl.Text = ofd.FileName;

ptb.ImageLocation = ofd.FileName;

}

}

//Đổi hình nền

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

{

new Wallpaper(txtUrl.Text).ChangeBackground();

}

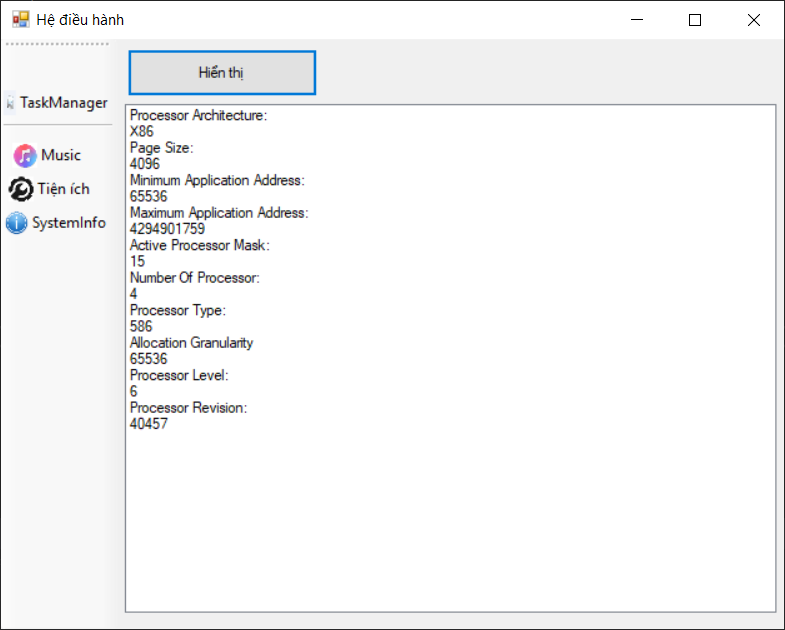
}

}

III/ Xem thông tin CPU

1. **Hình ảnh minh hoạ**

* Khi click vào nút “Hiển thị” sẽ liệt kê thông tin của CPU:

****

1. **Mã nguồn**

* ucSystemInfor:

using System;

using System.Collections;

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.Runtime.InteropServices;

namespace HeDieuHanh.GUI

{

public partial class ucSystemInfor : UserControl

{

[DllImport("kernel32.dll")]

public static extern void GetSystemInfo(out SystemInfo input); //Lấy thông tin CPU

//Kiểu dữ liệu tự định nghĩa về kiến trúc của CPU

public enum ProcessorArchitecture

{

X86 = 0,

X64 = 9,

@Arm = -1,

Itanium = 6,

Unknown = 0xFFFF,

}

[StructLayout(LayoutKind.Sequential)]

//Kiểu dữ liệu tổng hợp thông tin CPU

public struct SystemInfo

{

public ProcessorArchitecture ProcessorArchitecture; //Kiến trúc

public uint PageSize; // Kích cỡ trang

public IntPtr MinimumApplicationAddress; // Kích cỡ ứng dụng tối thiểu

public IntPtr MaximumApplicationAddress; // Kích cỡ ứng dụng tối đa

public IntPtr ActiveProcessorMask;

public uint NumberOfProcessors; //Số bộ xử lý

public uint ProcessorType; // Loại bộ xử lý

public uint AllocationGranularity;

public ushort ProcessorLevel;

public ushort ProcessorRevision;

}

public ucSystemInfor()

{

InitializeComponent();

}

// Thêm thông tin vào listbox

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

{

try

{

listBox1.Items.Clear();

SystemInfo pSI = new SystemInfo();

GetSystemInfo(out pSI);

string[] SystemInfoArray = new string[20];

SystemInfoArray[0] = "Processor Architecture:";

SystemInfoArray[1] = pSI.ProcessorArchitecture.ToString();

SystemInfoArray[2] = "Page Size:";

SystemInfoArray[3] = pSI.PageSize.ToString();

SystemInfoArray[4] = "Minimum Application Address:";

SystemInfoArray[5] = Convert.ToInt32(pSI.MinimumApplicationAddress.ToString("X"), 16).ToString();

SystemInfoArray[6] = "Maximum Application Address:";

SystemInfoArray[7] = Convert.ToInt64(pSI.MaximumApplicationAddress.ToString("X"), 16).ToString();

SystemInfoArray[8] = "Active Processor Mask:";

SystemInfoArray[9] = pSI.ActiveProcessorMask.ToString();

SystemInfoArray[10] = "Number Of Processor:";

SystemInfoArray[11] = pSI.NumberOfProcessors.ToString();

SystemInfoArray[12] = "Processor Type:";

SystemInfoArray[13] = pSI.ProcessorType.ToString();

SystemInfoArray[14] = "Allocation Granularity";

SystemInfoArray[15] = pSI.AllocationGranularity.ToString();

SystemInfoArray[16] = "Processor Level:";

SystemInfoArray[17] = pSI.ProcessorLevel.ToString();

SystemInfoArray[18] = "Processor Revision:";

SystemInfoArray[19] = pSI.ProcessorRevision.ToString();

listBox1.Items.AddRange(SystemInfoArray);

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

}

}

Mục lục

[Phần 1: Giới thiệu Windows API, cách sử dụng các hàm Windows API trong ngôn ngữ lập trình C# 1](#_Toc8832253)

[I/ Giới thiệu Windows API 1](#_Toc8832254)

[1. API là gì? 1](#_Toc8832255)

[2. Giới thiệu Windows API 1](#_Toc8832256)

[II/ Cách sử dụng hàm Windows API trong ngôn ngữ lập trình C# 2](#_Toc8832257)

[Phần 2: Viết ứng dụng sử dụng các hàm trong Windows API 3](#_Toc8832258)

[I/ Quản lý tiến trình 3](#_Toc8832259)

[II/ Các tiện ích 9](#_Toc8832260)

[A) Kiểm tra kết nối mạng 9](#_Toc8832261)

[B) Kiểm tra tình trạng pin 10](#_Toc8832262)

[C) Thay đổi âm lượng 12](#_Toc8832263)

[D) Thay đổi ngày giờ thiết bị 13](#_Toc8832264)

[E) Thay đổi hình nền 15](#_Toc8832265)

[F) Mã nguồn chung của tiện ích 17](#_Toc8832266)

[III/ Xem thông tin CPU 19](#_Toc8832267)