Trang bìa

Lúc nào text xong hết ở đây, thì paste trang bìa vào word ở máy local > **là xong**

MỤC LỤC

**MỞ ĐẦU**

1. **Giới thiệu**  
   Hệ điều hành là một hệ chương trình hoạt động giữa người dùng và phần cứng của máy tính. Từ đó, cung cấp một môi trường để người sử dụng có thể thi hành các chương trình làm cho máy tính dễ sử dụng hơn, thuận tiện và hiệu quả. Task Manager là một trong những tiện ích rất hữu ích được tích hợp trên hệ điều hành Windows. Chúng cho phép người dùng kiểm soát các chương trình đang chạy trên máy tính. Vì thế, nhóm chúng em đã quyết định lựa chọn đề tài: “**Xây dựng ứng dụng Task Manager sử dụng hàm API của Windows**”.
2. **Công cụ sử dụng**

Ngôn ngữ C#

Phần mềm code: VS 2019/2020

1. **Nội dung báo cáo:**

Nội dung báo cáo được trình bày 2 nội dung cơ bản

**Chương 1**: Tổng quan về hệ điều hành và hàm API trên hệ điều hành Windows

Chương này trình bày về các khái niệm tổng quát về hệ điều hành windows. Từ đó giới thiệu về tiến trình trong hệ điều hành windows.

**Chương 2:** Cách thức gọi hàm API bằng công cụ

Chương này trình bày về bài toán, cách thức chi tiết gọi hàm API bằng công cụ mà nhóm sử dụng

Ảnh chụp chương trình chạy và mã nguồn có đủ chú thích

Chương này trình bày về sản phẩm mà nhóm đã thực hiện

**Chương 3:** Kết luận

**Chương 1: Tổng quan về hệ điều hành và hàm API trên hệ điều hành Windows**

**1.1. Tổng quan về hệ điều hành windows**

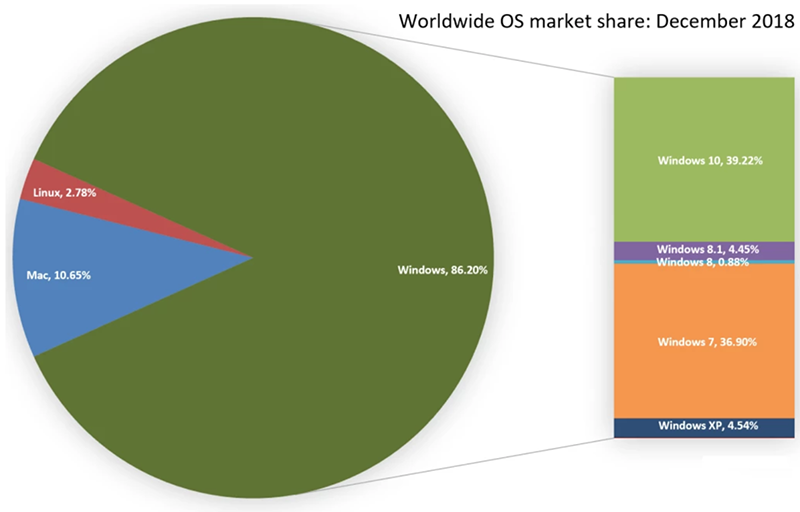
Hệ điều hành là một chương trình hay một hệ chương trình hoạt động giao

tiếp giữa người dùng và phần cứng máy tính. Cung cấp một môi trường để người sử dụng

có thể thi hành chương trình, giúp ích cho người dùng trong công việc và cuộc sống. Hệ

điều hành bao gồm các chương trình dịch, hệ thống cơ sở dữ liệu,... sử dụng tài nguyên

của máy tính để giải quyết các yêu cầu người dùng.



**Hình 1. Hình ảnh minh họa hệ điều hành Windows chiếm thị phần như thế nào hiện nay**

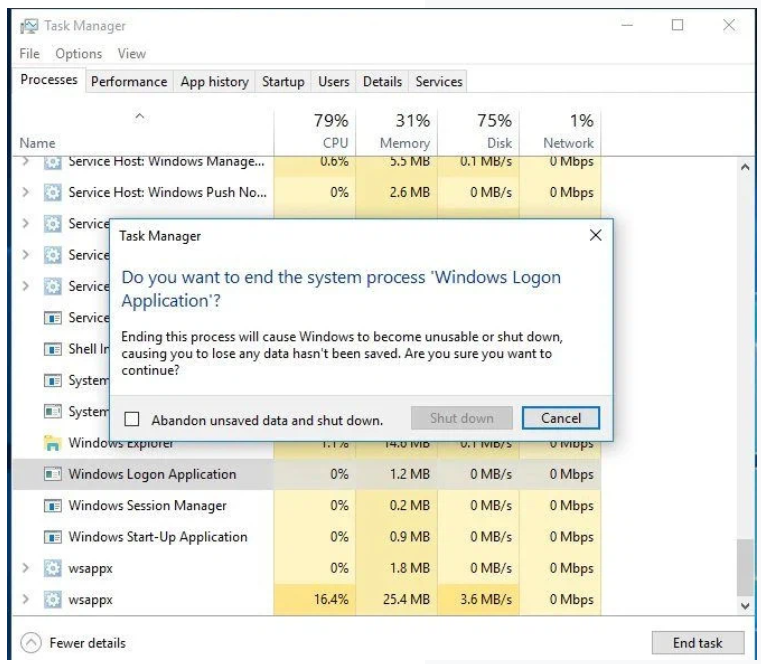
**1.2. Tổng quan về tiến trình**

**1.2.1. Tiến trình là gì?**

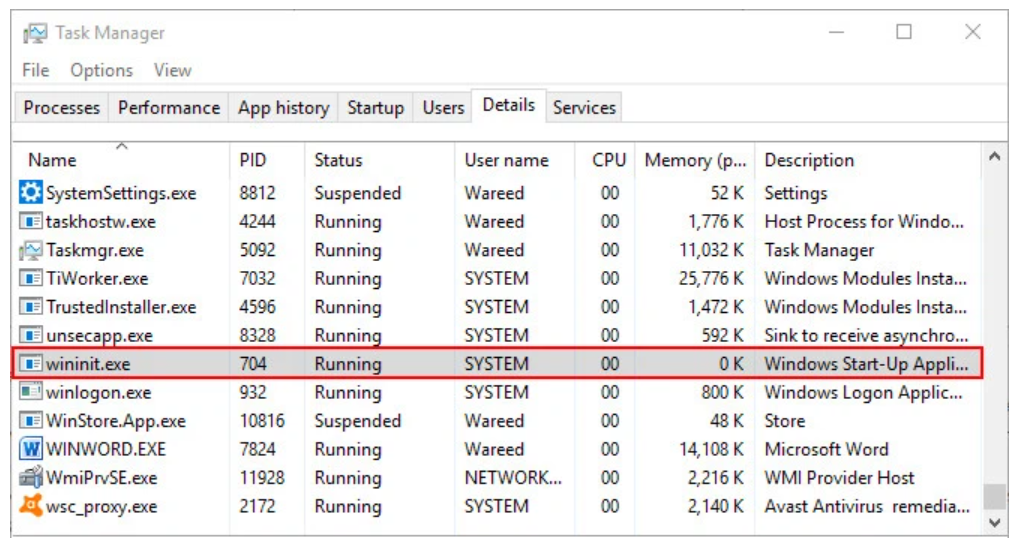
Tiến trình - process là một khái niệm cơ bản trong bất kì một hệ điều hành nào. Một tiến trình có thể được định nghĩa là một thực thể chương trình đang được chạy trong hệ thống. Một web server chạy trong thiết bị là một tiến trình, hoặc một chương trình soạn thảo văn bản đang chạy trong thiết bị cũng là một tiến trình.

**1.2.2. Ví dụ về các tiến trình thường sử dụng của windows**

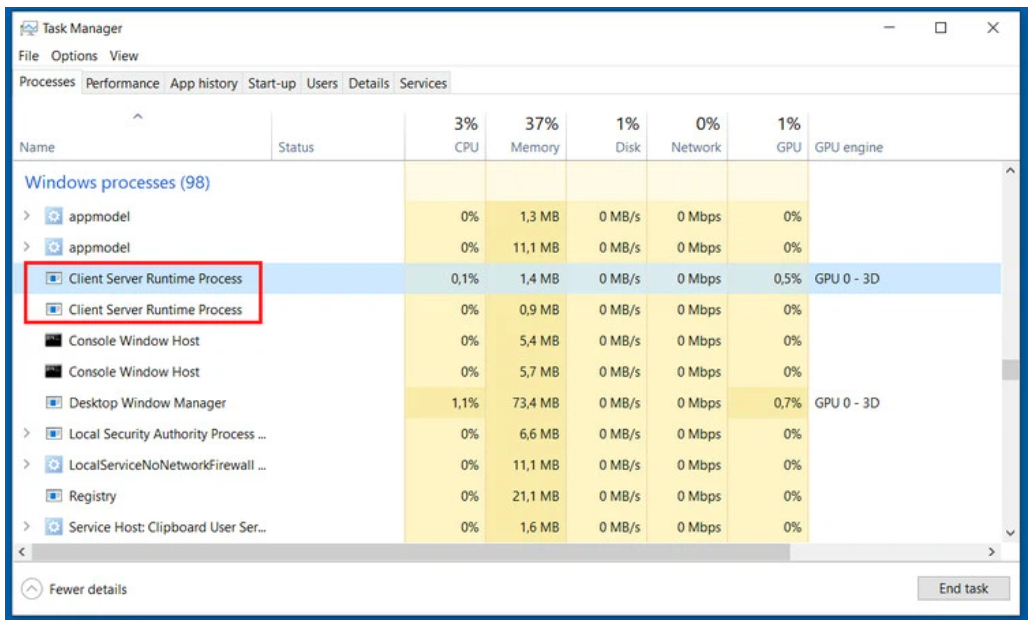
**1.2.2.1. Logon Application (winlogon.exe)**



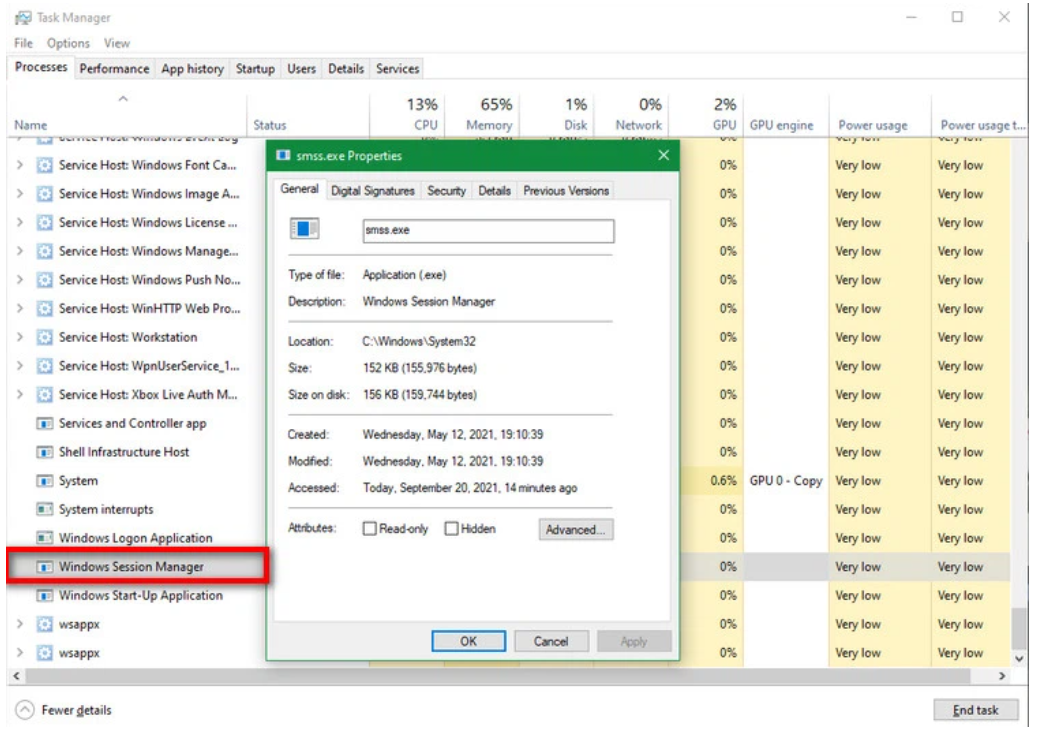
**1.2.2.2. Start-Up Application (wininit.exe)**



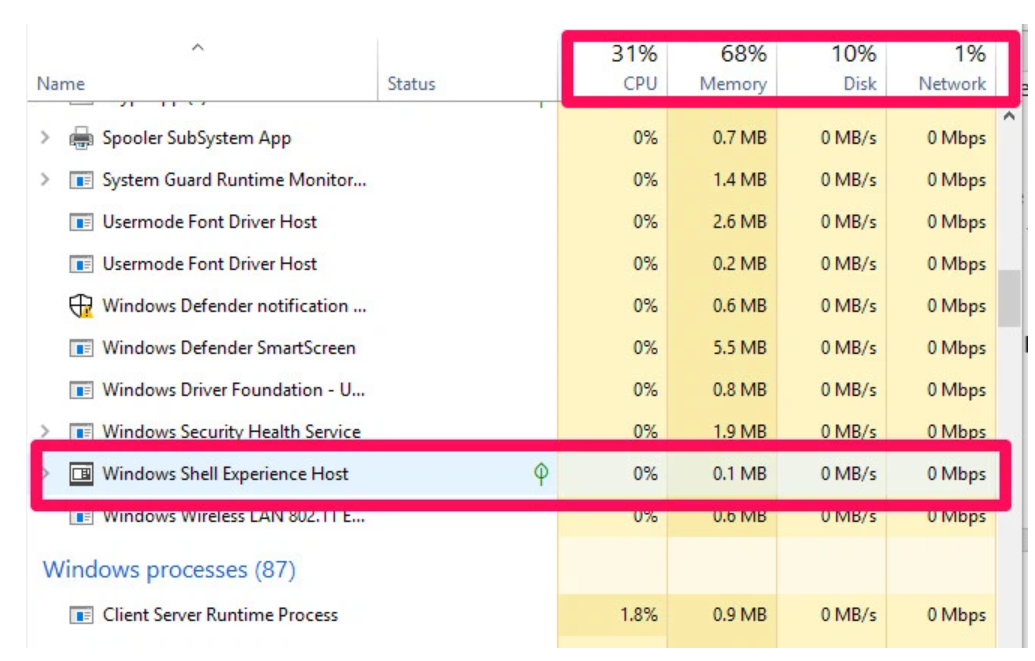
**1.2.2.3. Client Server Runtime Process (csrss.exe)**



**1.2.2.4. Session Manager (smss.exe)**



**1.2.2.5. Shell Experience Host**



**1.3 Hàm API trên hệ điều hành windows**

**1.3.1. Hàm API là gì?**

API là viết tắt của Application Programming Interface (giao diện lập trình ứng dụng)

phương thức kết nối với các thư viện và ứng dụng khác. Nó chính là một phần mềm giao

tiếp được sử dụng bởi các ứng dụng khác nhau giữa chương trình và hệ điều hành.

**1.3.2. Thành phần của Windows API**

Các hàm API có thể được chia thành một số loại chính:

* Base Services: thao tác với các tài nguyên trên Windows như hệ thống tập tin (file systems), các thiết bị, các tiến trình (processes), các luồng (threads), quản lý bộ nhớ,… Các hàm này nằm trong thư viện kernel32.dll.
* Advanced Services: truy xuất Windows registry, tắt, mở máy tính, Windows service, tài khoản người dùng,… Các hàm này nằm trong thư viện dvapi32.dll.
* Graphics Device Interface: xuất dữ liệu đồ họa ra màn hình, máy in,… Các hàm này nằm trong thư viện gdi32.dll.
* User Interface: cung cấp các thành phần giao diện người dùng như form, nút, textbox, nhận tín hiệu chuột, bàn phím,… Các hàm này nằm trong thư viện shell32.dll, user32.dll, comctl32.dll, comdlg32.dll:
  + Common Dialog Box Library: các hộp thoại cơ bản trong Windows mà hầu như ứng dụng nào cũng sử dụng, như hộp thoại mở file, lưu file,…
  + Common Control Library: nút, status bars, progress bars, toolbars, tabs,…
  + Windows Shell: thu tín hiệu chuột, bàn phím,
* Network Services: các thao tác liên quan đến mạng
* Multimedia: thao tác với các thiết bị đa phương tiện

**Chương 2: Cách thức gọi hàm API bằng công cụ**

**2.1. Giới thiệu bài toán chương trình ứng dụng**

* Chương trình Task Manager trên hệ điều hành Windows cho phép người dùng kiểm soát các chương trình đang chạy trên máy tính.
* Nguyên tắc chung của Task Manager là sử dụng hàm có sẵn

**2.2. Giải thuật của bài toán**

* Đầu vào:
* Người dùng mở ứng dụng Task Manager.
* Người dùng ấn chuột phải vào tiến trình hiện thời hiển thị trên danh sách hiển thị, chọn End task.
* Người dùng chọn options trên thanh menu, chọn Run new task
* Đầu ra:
* Ứng dụng hiển thị danh sách các tiến trình hiện thời đang chạy trên máy tính của người dùng.
* Ứng dụng thực hiện xử lý kết thúc tác vụ đó trên máy tính người dùng.
* Ứng dụng mở ra 1 form mới, cho phép người dùng nhập tác vụ mới, nhấn Run để mở tác vụ đó trên máy tính

**Các bước chi tiết:**

**2.2.1. Hiển thị danh sách các tiến trình đang chạy**

* Đầu vào:
* Người dùng mở ứng dụng Task Manager.
* Đầu ra:
* Ứng dụng hiển thị các tiến trình hiện thời đang chạy trên máy tính của người dùng.

Bước 1: Xử lý lấy các tiến trình đang chạy trên Window

Bước 2: Xử lý trả về các tiến trình đang chạy đó

Bước 3: Xử lý hiển thị các tiến trình đang chạy bao gồm tên và bộ nhớ chiếm giữ

**2.2.2. Kết thúc một tiến trình đang chạy**

* Đầu vào:
* Người dùng ấn chuột phải vào tiến trình hiện thời hiển thị trên danh sách hiển thị, chọn End task.
* Đầu ra:
* Ứng dụng thực hiện xử lý kết thúc tác vụ đó trên máy tính người dùng.

Bước 1: Bắt sự kiện người dùng chọn end task

Bước 2: Xử lý kết thúc tiến trình được chọn

Bước 3: Xử lý hiển thị các tiến trình đang chạy khác

**2.2.3. Mở một tiến trình mới**

* Đầu vào:
* Người dùng chọn options trên thanh menu, chọn Run new task
* Đầu ra:
* Ứng dụng mở ra 1 form mới, cho phép người dùng nhập tác vụ mới, nhấn Run để mở tác vụ đó trên máy tính

Bước 1: Bắt sự kiện người dùng chọn chức năng Run new task

Bước 2: Xử lý lấy tên tiến trình được nhập

Bước 3: Xử lý chạy tiến trình đó

Bước 4: Xử lý trả về tiến trình đang chạy

Bước 5: Xử lý hiển thị tiến trình đang chạy

**2.3. Mã nguồn chương trình ứng dụng và thực nghiệm**

**2.3.1. Mã nguồn chương trình**

**2.3.1.1. Form.cs**

* **Khai báo thư viện:**

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; //Thư viện cho phép truy cập các tiến trình

namespace BTL\_HĐH

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

// khi load form lên thì gọi tới hàm getprocess khai báo ở dưới

GetProcess();

}

* Khởi tạo mảng chứa các tiến trình sẽ được lấy

//Danh sách các Process đang chạy > gán vào 1 mảng tên process

Process[] process;

* **Xử lý gọi các tiến trình đang chạy**

void GetProcess()

{

// dùng hàm GetProcesses() > gọi các tiến trình hiện tại đang chạy > trả về 1 mảng kết quả > gán vào biến process khai báo ngoài

process = Process.GetProcesses();

// clear các item của listview hiển thị sau mỗi lần load mới dữ liệu

listView1.Items.Clear();

* **Xử lý hiển thị các tiến trình đang chạy ra ListView**

// duyệt từng item trong mảng process vừa get

foreach (var item in process) {

//Tên process đẩy vào trường text, gán object đó vào biến NewItem

ListViewItem NewItem = new ListViewItem() { Text = item.ProcessName };

// add 1 subitem (thêm vào object) 1 thuộc tính : memorysize

NewItem.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = item.PagedSystemMemorySize.ToString() });

// đẩy giá trị trong biến NewItem vào lần lượt các items hiển thị trên listview (gồm processname và page system memory size)

listView1.Items.Add(NewItem);

}

}

* **Xử lý load lại danh sách khi thêm hoặc kết thúc tiến trình, cập nhật theo thời gian**

//Load lại và cập nhật danh sách process

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

{

// nếu get danh sách hiện thời có thêm hoặc ít hơn danh sách cũ thì load lại

if (process.Length != Process.GetProcesses().Length)

{

GetProcess(); // gọi lại hàm getprocess > show lên listview hiện tại

}

}

* **Xử lý kết thúc tiến trình đang chạy**

//Đóng process đang chạy > chuột phải chọn endtask

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

{

// bắt sự kiện nếu: item được select có số count > 0 thì:

if(listView1.SelectedItems.Count > 0)

{

// tạo 1 biến index để lưu chỉ mục của tiến trình được chọn

int index = 0;

// duyệt tất cả các item 1 trong process get hiện thời

foreach (var item in process)

{

// nếu tên của item của mảng hiện thời trùng với text của item được select trong listview thì:

if(item.ProcessName == listView1.SelectedItems[0].Text)

{

// gán index của item được chọn trong process đó vào biến index

index = process.ToList().IndexOf(item);

}

}

// thoát khỏi vòng foreach > lấy được index > gọi hàm Kill() để end process[thứ index vừa lấy được]

process[index].Kill();

}

}

//Chạy process mới > khi mở menu strip lựa chọn run new task

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

{

// khi bấm vào lựa chọn ở menu > mở ra form mới có tên là AddProcess()

using (AddProcess frm = new AddProcess())

{

// nếu user chọn OK > gọi hàm GetProcess() > hiển thị cái item process lên listview

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

{

GetProcess();

}

}

}

}

}

**2.3.1.1. AddProcess.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;

namespace BTL\_HĐH

{

public partial class AddProcess : Form

{

public AddProcess()

{

InitializeComponent();

}

* **Xử lý chạy một tiến trình mới**

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

{

// kiểm tra: textbox nhập text của tiến trình khác rỗng, thì:

if (!string.IsNullOrEmpty(textProcessName.Text))

{

try

{

// tạo 1 proc mới

Process proc = new Process();

// gán proc đó với file name là text của process vừa nhập

proc.StartInfo.FileName = textProcessName.Text;

// start proc đó, với file name gán ở trên

proc.Start();

}

catch(Exception ex)

{

// catch lỗi

MessageBox.Show(ex.Message, "Hãy nhập tên process", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

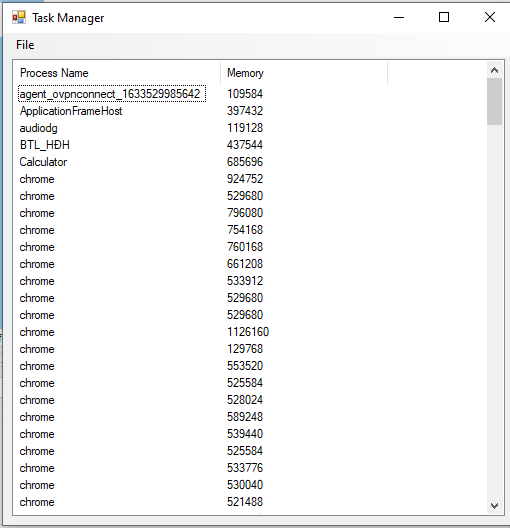
}

}

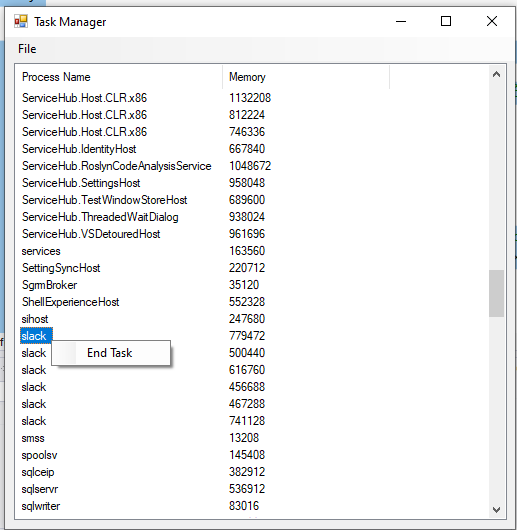
}

**2.3.2. Hình ảnh màn hình thực nghiệm chương trình**

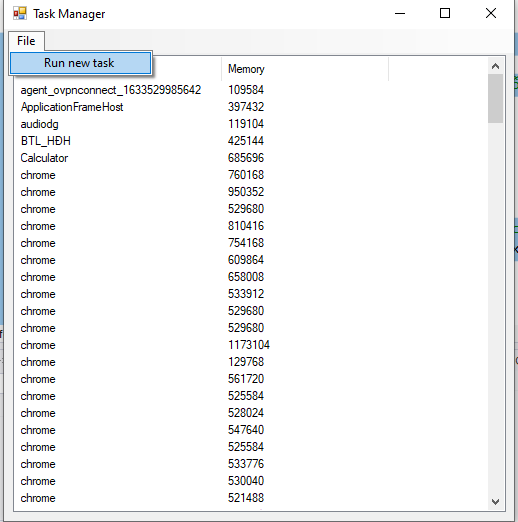
**2.3.2.1. Mở task manager**

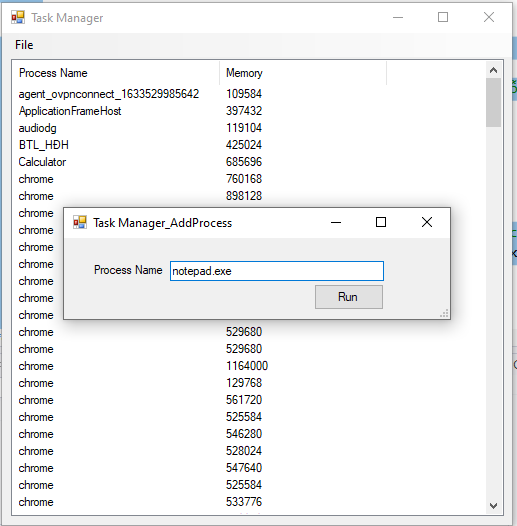


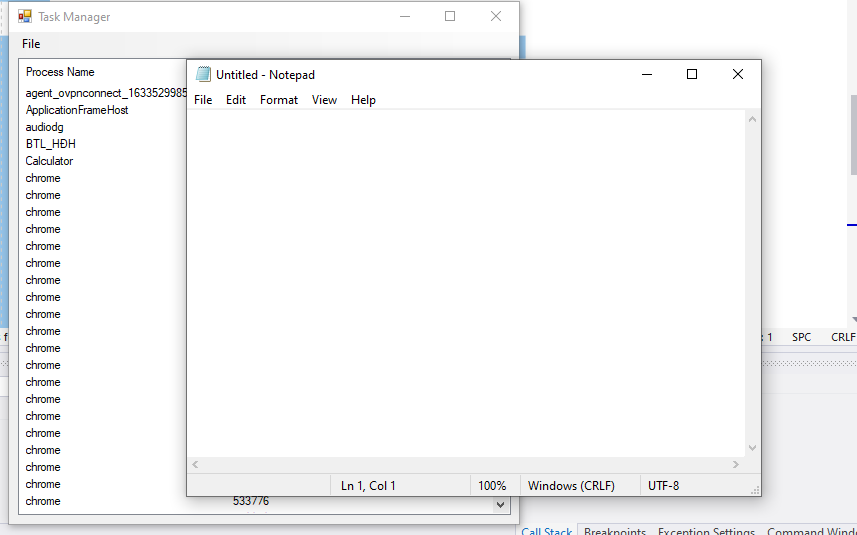
**2.3.2.2. End task**



**2.3.2.3. Run new task**







**Chương 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

3.1. Kết luận

Để có thể quản lý tốt các tiến trình đang hoạt động trên máy, chúng ta cần những ứng dụng như task manager.

Sau thời gian nghiên cứu học hỏi, nhóm em đã xây dựng thành công chương trình

ứng dụng hỗ trợ task manager cho hệ điều hành windows. Với chức năng run, end các tiến trình của hệ điều hành. Tuy nhiên chương trình không tránh được nhiều sai sót,

nhóm mình rất mong nhận được góp ý từ thầy. Nhóm em chân thành cảm ơn thầy nhiều.

3.2. Hướng phát triển

Chúng em sẽ cố gắng hoàn thiện ứng dụng Task Manager trong thời gian tới với các chức năng tương tự như task manager của window.

Chúng em xin cảm ơn thầy trong suốt quá trình làm bài tập lớn đã nhiệt tình chỉ dẫn, hỗ trợ bọn em để bọn em có thể hoàn thành bài tập lớn lần này.