**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----- & -----**

****

**BÁO CÁO BÀI TẬP LỚN MÔN HỌC**

**HỆ ĐIỀU HÀNH**

|  |
| --- |
| ***Đề tài:* Chương trình ứng dụng quản lý tiến trình**    **Nhóm thực hiện:**  **Thành viên:** Nguyễn Văn Mạnh  Mai Tiến Đạt  Đỗ Thị Thanh Phương  Nguyễn Hải Quân  Trần Huy Nam  Hà Nội, ngày 24 tháng 04 năm 2022 |

# MỤC LỤC

[MỤC LỤC 2](#_Toc101731214)

[MỞ ĐẦU 3](#_Toc101731215)

[1. Giới thiệu: 3](#_Toc101731216)

[2. Công cụ sử dụng: 3](#_Toc101731217)

[3.Nội dung bố cục của báo cáo 3](#_Toc101731218)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH VÀ HÀM API TRÊN HỆ ĐIỀU HÀNH WINDOWS 4](#_Toc101731219)

[1.1 . Tổng quan về hệ điều hành windows 4](#_Toc101731220)

[1.2 Hàm API trên hệ điều hành windows 4](#_Toc101731221)

[1.2.1 Hàm API là gì? 4](#_Toc101731222)

[1.2.2 Một số hàm API trên hệ điều hành windows 4](#_Toc101731223)

[CHƯƠNG 2: CHƯƠNG TRÌNH ỨNG DỤNG HỖ TRỢ HỆ ĐIỀU HÀNH VÀ THỰC NGHIỆM 6](#_Toc101731224)

[2.1 Giới thiệu bài toán chương trình ứng dụng. 6](#_Toc101731225)

[2.2 Giải thuật của bài toán. 6](#_Toc101731226)

[2.3 Mã nguồn chương trình ứng dụng và thực nghiệm. 6](#_Toc101731227)

[2.3.1 Mã nguồn chương trình 6](#_Toc101731228)

[2.3.2 Hình ảnh màn hình thực nghiệm chương trình 18](#_Toc101731229)

[CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 20](#_Toc101731230)

[3.1 Kết luận 20](#_Toc101731231)

[3.2 Hướng phát triển 20](#_Toc101731232)

[TÀI LIỆU THAM KHẢO 21](#_Toc101731233)

# 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. Mỗi máy tính đều có các trình quản lý các tiến trình. Nên để có thể giúp người dùng xem và ngắt các tiến trình của hệ thống, nhóm đã quyết định lựa chọn đề tài “**Xây dựng ứng dụng xử lý tiến trình hệ thống hỗ trợ hệ điều hành windows**”.

## 2. Công cụ sử dụng:

Ngôn ngữ lập trình: C++.

Phần mềm code: Visual studio 2019.

## 3.Nội dung bố cục của báo cáo

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

Chương này trình bày các khái niệm tổng quan logic về hệ điều hành windows. Từ đó, giới thiệu về một số hàm API và công dụng của hàm API trên windows.

**Chương 2:** Chương trình ứng dụng ngôn ngữ hỗ trợ hệ điều hành và thực nghiệm.

Chương này, trình bày bài toán, giải thuật ứng dụng gọi ngắt tiến trình.

**Chương 3:** Kết luận và hướng phát triển.

Chương này đưa ra kết luận về những gì đã làm được và rút ra bài học định hướng phát triển ứng dụng cho ứng dụng hỗ trợ hệ điều hành.

# 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

Định nghĩa: 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.

Hệ điều hành windows: Một hệ điều hành đa nhiệm có thể xử lý nhiều chương trình cùng một lúc. Hệ điều hành được phân phối bởi Microsoft.

## 1.2 Hàm API trên hệ điều hành windows

### 1.2.1 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.

### 1.2.2 Một số hàm API trên hệ điều hành windows

- Các API được tổ chức trong 4 DLL của windows:

1. **KERNEL32**:

Là DLL chính, đảm nhiệm quản lý bộ nhớ, thực hiện chức năng đa nhiệm và những hàm ảnh hưởng trực tiếp đến hoạt động của Windows.

1. **USER32**:

Thư viện quản lý Windows. Thư viện này chứa các hàm xử lý menu, định giờ, truyền tin, tập tin và nhiều phần không được hiển thị khác của Windows.

1. **GDI32**:

Giao diện thiết bị đồ họa (Graphics Device Interface). Thư viện này cung cấp các hàm vẽ trên màn hình, cũng như kiểm tra phần biểu mẫu nào cần vẽ lại.

1. **WINNM**:

Cung cấp các hàm multimedia để xử lý âm thanh, nhạc, video thời gian thực, lấy mẫu, v.v… Nó là DLL 32 bit. (Thư viện 16 bit tên làMMSYSTEM)

Ta có thể tìm các tập tin này trong thư mục \Windows\system. Ngoài ra, còn có các DLL nhỏ hơn, cũng được dùng phổ biến để cung cấp các dịch vụ đặc biệt cho ứng dụng.

Trên đây là các tên DLL 32 bit. Phiên bản VB4 là bản cuối cùng còn hỗ trợ 16 bit.

# CHƯƠNG 2: CHƯƠNG TRÌNH ỨNG DỤNG HỖ TRỢ HỆ ĐIỀU HÀNH VÀ THỰC NGHIỆM

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

\* Chương trình xử lý ngắt tiến trình trên Hệ điều hành windows hỗ trợ người dùng khi sử dụng máy tính.

\* Sử dụng để hiển thị và ngắt các tiến trình đang chạy trên hệ điều hành windows

thiết bị và chương trình ghi.

Từ đó, chương trình sẽ sử dụng hỗ trợ hàm API trên hệ điều hành windows.

## 2.2 Giải thuật của bài toán.

**Đầu vào :**Tiến trình của hệ điều hành windows

**Đầu ra :** Cho phép người dùng xem các tiến trình trên thiết bị và có thể dừng tiến trình.

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

### 2.3.1 Mã nguồn chương trình

#include <windows.h>

#include <TlHelp32.h>

#include <stdio.h>

#include <commctrl.h>

LRESULT CALLBACK WindowProcedure(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam);//Lresult là một giá trị nguyên mà chương trình của bạn trở lại WINDOW chứa phản hồi từ chương trình cho một tin nhắn cụ thể (mã thông báo)

Có 4 tham số:

hwnd là một handle tới window.

message là mã message.

wParam and lParam chứa dữ liệu bổ sung liên quan đến message.

char szClassName[] = "ProcessMngr";

int x = GetSystemMetrics(SM\_CXSCREEN) / 3,// chiều rộng của màn hình hiển thị chính / 3

y = GetSystemMetrics(SM\_CYSCREEN) / 3;// chiều cao của màn hình hiển thị chính / 3

int iSelect;

int \_procID;

HWND proc\_list;

LPWSTR \_processName;//L:Long// P:Pointer//W:Wide(Trong c++ thì wide được định nghĩa cho ký tự unicode)//STR:String

wchar\_t \_processID[10];//Kiểu dữ liệu

wchar\_t data[10];//Kiểu dữ liệu

char \_buffProcessID[10];//Kiểu dữ liệu

HMENU hPopupMenu;// menu

POINT p;

* hàm hiển thị tiến trình

int \_showProcesses();

* hàm ngắt tiến trình

BOOL \_killProcess(int \_procID);

* hàm thêm cột

void AddColumn(HWND hView, LPSTR text, int Col, int Width, DWORD dStyle);

* hàm thêm index

void AddIndex(HWND hView, int Index);

* hàm thêm dòng

void AddRow(HWND hView, LPWSTR text, int Index, int Col);

* hàm tự động làm mới các tiến trình

DWORD WINAPI \_autoRefresh(LPVOID);

* CreateFont() tạo một HFONT, một handle tới một Logical Font trong bộ nhớ.

HFONT hFont = CreateFont(14, 0, 0, 0, FW\_DONTCARE, FALSE, FALSE, FALSE, ANSI\_CHARSET, OUT\_TT\_PRECIS, CLIP\_TT\_ALWAYS, DEFAULT\_QUALITY, FF\_DONTCARE, L"Arial");

* hàm winmain hiển thị ra window

int WINAPI WinMain(HINSTANCE hThisInstance, //Quản lý việc thực thi của chương trình, nó quản lý tài nguyên của chương trình đang cài đặt

HINSTANCE hPrevInstance, //Con trỏ trỏ đến việc thực thi trước đó

LPSTR lpszArgument, //Command line arguments, chuỗi ký tự bình thường

int iCmdShow) //1 số nguyên để cung cấp cho hàm ShowWindow()

{

HWND hwnd; //handle window

MSG messages; //chứa thông tin message

WNDCLASSEX wincl; //chứa thông tin window class, được dùng với RegisterClassEx

wincl.hInstance = hThisInstance;//Thiết lập phiên bản

wincl.lpszClassName = (LPCWSTR)szClassName;//Tên của window class

wincl.lpfnWndProc = WindowProcedure;//trỏ đến WindowProcedure

wincl.style = CS\_DBLCLKS; kiểu của class

wincl.cbSize = sizeof(WNDCLASSEX); //chỉ định kích thước của cấu trúc

wincl.hIcon = LoadIcon(NULL, IDI\_APPLICATION);//Tải biểu tượng mặc định

wincl.hIconSm = LoadIcon(NULL, IDI\_APPLICATION); icon nhỏ của class

wincl.hCursor = LoadCursor(NULL, IDC\_ARROW);//Tải con trỏ mũi tên

wincl.lpszMenuName = NULL;//Không cần một menu

wincl.cbClsExtra = 0; // không có bộ nhớ class bổ sung

wincl.cbWndExtra = 0; // không có bộ nhớ window bổ sung

wincl.hbrBackground = (HBRUSH)COLOR\_BACKGROUND;// màu background

if (!RegisterClassEx(&wincl)) // đăng ký 1 window class để sử dụng cho CreateWindowEx

return 0;

hwnd = CreateWindowEx( //Tạo một cửa sổ con,cửa sổ bật lên hoặc cửa sổ con trồng lên nhau với kiểu cửa sổ mở rộng hoặc giống như create window(tạo một cửa sổ con)

WS\_EX\_CLIENTEDGE | WS\_EX\_DLGMODALFRAME,

(LPCWSTR)szClassName,

L"Process Manager",

WS\_SYSMENU | WS\_MINIMIZEBOX,

x,

y,

325,

390,

HWND\_DESKTOP,

NULL,

hThisInstance,

NULL

);

ShowWindow(hwnd, iCmdShow); // đặt trạng thái hiển thị của window được chỉ định.

while (GetMessage(&messages, NULL, 0, 0))

{

TranslateMessage(&messages);//Chuyển các tin nhắn phím ảo thành các tin nhắn ký tự. Các thông điệp ký tự được đăng vào hàng đợi tin nhắn của chuỗi gọi, sẽ được đọc vào lần tiếp theo chuỗi gọi hàm [GetMessage](https://docs.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-getmessage)

DispatchMessage(&messages);//Gửi một thông báo đến một thủ tục của window.Nó thường được sử dụng để gửi một tin nhắn được truy xuất bởi chức năng GetMessage

}

return messages.wParam;// trả về wParam của message

}

LRESULT CALLBACK WindowProcedure(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)//gọi thủ tục thông báo của window//HWND là xử lý cho window//UNIT là mã nhắn tin//wparam//lparam chứa dữ liệu bổ sung liên quan đến tin nhắn

{

switch (message) // bắt các trường hợp của message

{

case WM\_DESTROY://nó được gửi đến thủ tục cửa sổ của windows khi một cửa sổ bị hủy

PostQuitMessage(0);//Cho hệ thống biết rằng một luồng đã đưa ra yêu cầu kết thúc (thoát)

break;

case WM\_CREATE://Khởi tạo cửa sổ

InitCommonControls();//Đăng ký và khởi tạo một số lớp cửa sổ điều khiển chung nhất định

proc\_list = CreateWindowEx(

0,

WC\_LISTVIEW,

0,

WS\_CHILD | WS\_VISIBLE | LVS\_REPORT | LVS\_SINGLESEL | WS\_BORDER,

0,

0,

306,

349,

hwnd,

NULL,

0,

NULL

);

SendMessage(proc\_list, WM\_SETFONT, (WPARAM)hFont, 0);//Gửi thông báo được chỉ định đến một cửa sổ hoặc các cửa sổ. Hàm SendMessage gọi thủ tục cửa sổ cho cửa sổ được chỉ định và không trả về cho đến khi thủ tục cửa sổ xử lý thông báo.

ListView\_SetExtendedListViewStyle(proc\_list, LVS\_EX\_FULLROWSELECT);//Dùng để sử dụng điều khiển chế độ xem danh sách mở rộng

AddColumn(proc\_list, (LPSTR)L"Name", 0, 230, LVCFMT\_LEFT);//Thêm cột Name

AddColumn(proc\_list, (LPSTR)L"ID", 1, 70, LVCFMT\_LEFT);//Thêm cột ID

\_showProcesses();

CreateThread(NULL, 0, \_autoRefresh, 0, 0, NULL);//Tạo một chuỗi để thực thi trong không gian địa chỉ ảo của quá trình gọi.

break;

case WM\_NOTIFY: // message được gửi từ control

if (((LPNMHDR)lParam)->hwndFrom == proc\_list && ((LPNMHDR)lParam)->code == NM\_RCLICK) nếu message là tiến trình và click chuột phải

{

iSelect = SendMessage(proc\_list, LVM\_GETNEXTITEM, -1, LVNI\_FOCUSED);//Gửi thông báo được chỉ định đến một cửa sổ hoặc các cửa sổ. Hàm SendMessage gọi thủ tục cửa sổ cho cửa sổ được chỉ định và không trả về cho đến khi thủ tục cửa sổ xử lý thông báo.

ListView\_GetItemText(proc\_list, iSelect, 1, data, sizeof(data))//Nhận văn bản hoặc trang con ở chế độ xem danh sách

\_procID = \_wtoi(data);//Chuyển đổi một chuỗi thành số nguyên

if (iSelect >= 0) nếu có chọn tiến trình

{

GetCursorPos(&p);//Truy xuất vị trí con trỏ trên vị trí màn hình

hPopupMenu = CreatePopupMenu();//Tạo menu thả xuống, menu con

InsertMenu(hPopupMenu, 0, MF\_BYPOSITION | MF\_STRING, 1, L"Terminate");//Chèn một mục menu mới vào menu, di chuyển các mục khác xuống menu.

int choice = TrackPopupMenu(hPopupMenu, TPM\_BOTTOMALIGN | TPM\_LEFTALIGN | TPM\_RETURNCMD, p.x, p.y + 23, 0, hwnd, NULL);//Hiển thị menu lối tắt tại vị trí đã chỉ định và theo dõi việc lựa chọn các mục trên menu.Menu lối tắt xuất hiện ở bất kỳ đâu trên màn hình

switch (choice) //kiểm tra nếu chọn vào mục ở trên menu thì sẽ gọi đến hàm \_killProcess

{

case 1:

\_killProcess(\_procID);

break;

}

DestroyMenu(hPopupMenu); //Huỷ menu và giải phóng bộ nhớ mà menu chiếm

}

ListView\_SetItemState(proc\_list, iSelect, FALSE, LVIS\_FOCUSED);//Thay đổi trạng thái của một mục trong điều khiển dạng xem danh sách

}

return false;

default:

return DefWindowProc(hwnd, message, wParam, lParam);Hàm này như một phần của quá trình xử lý tin nhắn trong trường hợp như vậy ứng dụng có thể sửa đổi các tham số thông báo trước khi chuyển thông báo tới DefWindowProc

}

return 0;

}

int \_showProcesses()

{

ListView\_DeleteAllItems(proc\_list);//Xóa tất cả các danh sách tiến trình

typedef BOOL(WINAPI\* TH32\_PROCESS)//Thêm tên mới vào kiểu có sẵn

(HANDLE hSnapShot, LPPROCESSENTRY32 lppe);

static TH32\_PROCESS pProcess32First = NULL;//static là khai báo biến chung áp dụng chung cho đối tượng

static TH32\_PROCESS pProcess32Next = NULL;

PROCESSENTRY32 pe32 = { 0 };// danh sách các tiến trình

HANDLE hSnapshot = NULL;

HINSTANCE hDll = LoadLibrary(L"kernel32.dll"); xử lý cho hDll//LoadLibrary tải lên các thư viện của Window//kernel32.dll Tạo thành mã cơ bản mà được xây dựng và cung cấp các chức năng cơ bản như quản lý bộ nhớ,xử lý ngắt vào ra chuyển chúng thành các lệnh CPU và hướng dẫn cho các thành phần điện tử

if (hDll == 0)

{

MessageBox(NULL, L"Error getting processes !", L"Error", MB\_OK | MB\_ICONERROR);Hiển thị thông báo lỗi

return 0;

}

pProcess32First = (TH32\_PROCESS)GetProcAddress(hDll, "Process32First");//Truy xuất về thông tin quy trình đầu tiên

pProcess32Next = (TH32\_PROCESS)GetProcAddress(hDll, "Process32Next");//Truy xuất thông tin quy trình tiếp theo

hSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);// chụp lại quy trình được chỉ định,cũng như moodun và luồng được sử dụng bởi quy trình này

if (hSnapshot != (HANDLE)-1)

{

pe32.dwSize = sizeof(PROCESSENTRY32);//Đặt kích thước cấu trúc trước khi sử dụng

if (pProcess32First(hSnapshot, &pe32))

{

while (pProcess32Next(hSnapshot, &pe32))

{

\_processName = new wchar\_t[pe32.dwSize];

sprintf(\_buffProcessID, "%d", pe32.th32ProcessID); // ghi vào dạng chuỗi

mbstowcs(\_processID, \_buffProcessID, 10); // chuyển đổi sang chuỗi ký tự

MultiByteToWideChar(CP\_ACP, 0, (LPCSTR)pe32.szExeFile, pe32.dwSize, \_processName, pe32.dwSize);//MultiByteToWideChar là Ánh xạ một chuỗi ký tự thành một chuỗi UTF-16(Ký tự rộng) chuỗi ký tự không nhất thiết phải từ một bộ ký tự nhiều byte

AddIndex(proc\_list, 0);Thêm thuộc tính vào chỉ mục được sử dụng để tìm kiếm

AddRow(proc\_list, \_processName, 0, 0);//Thêm một hàng mới vào cuối giá trị cột

AddRow(proc\_list, \_processID, 0, 1);//Thêm một hàng mới vào cuối giá

AddRow(proc\_list, \_processName, 0, 0);//Thêm một hàng mới vào cuối giá

}

}

}

CloseHandle(hSnapshot);

return 0;

}

BOOL \_killProcess(int procID)

{

HANDLE hProcessSnap;//Tìm địa chỉ của window trong bộ nhớ

HANDLE hProcess;

PROCESSENTRY32 pe32;//Kiểu dữ liệu

hProcessSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

//Hàm CreateToolhelp32Snapshot chụp nhanh các quy trình chỉ định ,modun được sử dụng bời quy trinh này

pe32.dwSize = sizeof(PROCESSENTRY32);//Chứa kích thước của bộ đệm màn hình //Nhận một tham số bất kỳ kiểu dữ liệu nào trả về kích thước của kiểu dữ liệu đó

Process32First(hProcessSnap, &pe32);//Truy xuất thông tin đầu tiên gặp phải trong hệ thống

while (Process32Next(hProcessSnap, &pe32))//Hiển thị lên các quy trình trên hệ thống

{

if (procID == pe32.th32ProcessID)

{

hProcess = OpenProcess(PROCESS\_TERMINATE, 0, pe32.th32ProcessID);//Hàm OpenProcess trả về xử lý của tiến trình //Bắt buộc phải kết thúc một quá trình bằng Terminalprocess(Chấm dứt quá trình được chỉ định và tất cả luồng của nó)

TerminateProcess(hProcess, 0);//Chấm dứt quá trình được chỉ định và tất cả các luồng của nó.

ListView\_DeleteItem(proc\_list, iSelect); //Xóa một mục khỏi một danh sách điều khiển }

}

CloseHandle(hProcessSnap);//Đóng một xử lý

CloseHandle(hProcess);

return 0;

}

DWORD WINAPI \_autoRefresh(LPVOID)//Dword là khai báo kiểu số nguyên không dấu//WINAPI là hàm gọi chức năng của API Window

{

while (1)

{

Sleep(7500);

\_showProcesses();

}

}

void AddColumn(HWND hView, LPSTR text, int Col, int Width, DWORD dStyle)//Thêm cột và chuyển chế độ chỉnh sửa

{

LVCOLUMN lvc = { 0 };//Chứa thông tin về một cột trong chế độ xem báo cáo. Cấu trúc này được sử dụng cho cả việc tạo và thao tác với các cột

memset(&lvc, 0, sizeof(LVCOLUMN));

lvc.mask = LVCF\_TEXT | LVCF\_SUBITEM | LVCF\_WIDTH | LVCF\_FMT;

lvc.fmt = dStyle;

lvc.iSubItem = Col;

lvc.cx = Width;

lvc.pszText = (LPWSTR)text;

ListView\_InsertColumn(hView, Col, &lvc);//Chèn một cột mới trong điều khiển chế độ xem danh sách

}

void AddIndex(HWND hView, int Index)

{

LVITEM lv; Chỉ định nhận các thuộc tính của một mục dạng xem danh sách .cấu trúc này đã được cập nhật mới để hỗ trợ giá trị mặt nạ mới cho phép thụt lề mục

memset(&lv, 0, sizeof(LVITEM)); //lấp đầy một khối bộ nhớ với một giá trị cụ thể

lv.mask = LVIF\_TEXT;

lv.iSubItem = 0;

lv.pszText = LPSTR\_TEXTCALLBACK;

lv.iItem = Index;

ListView\_InsertItem(proc\_list, &lv);//chèn một mục mới vào danh sách

}

void AddRow(HWND hView, LPWSTR text, int Index, int Col)//Thêm một dòng trống vào bảng.

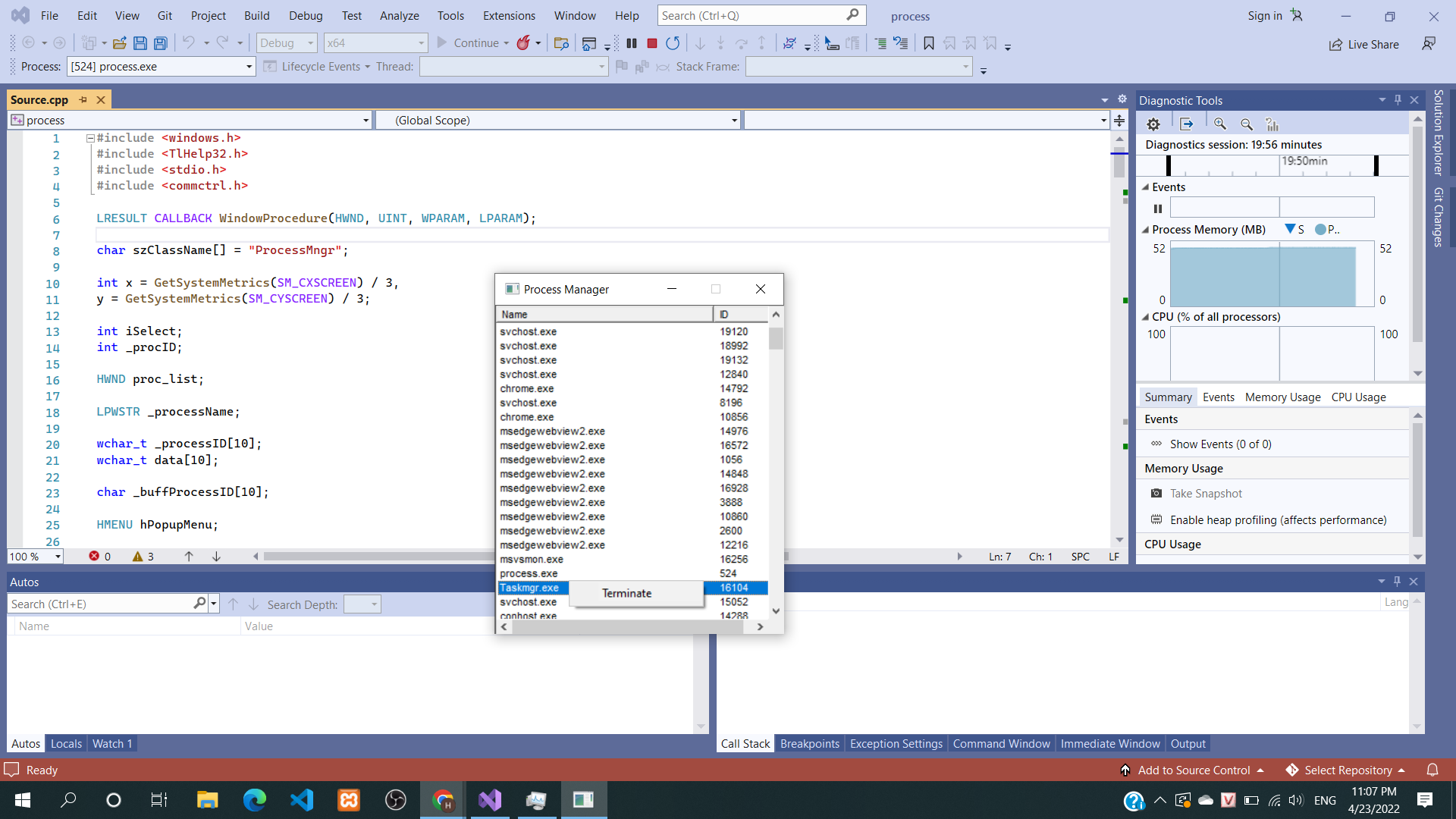
{

ListView\_SetItemText(hView, Index, Col, (LPWSTR)text);//Thay đổi văn bản của một mục hoặc trang con ở chế độ xem danh sách

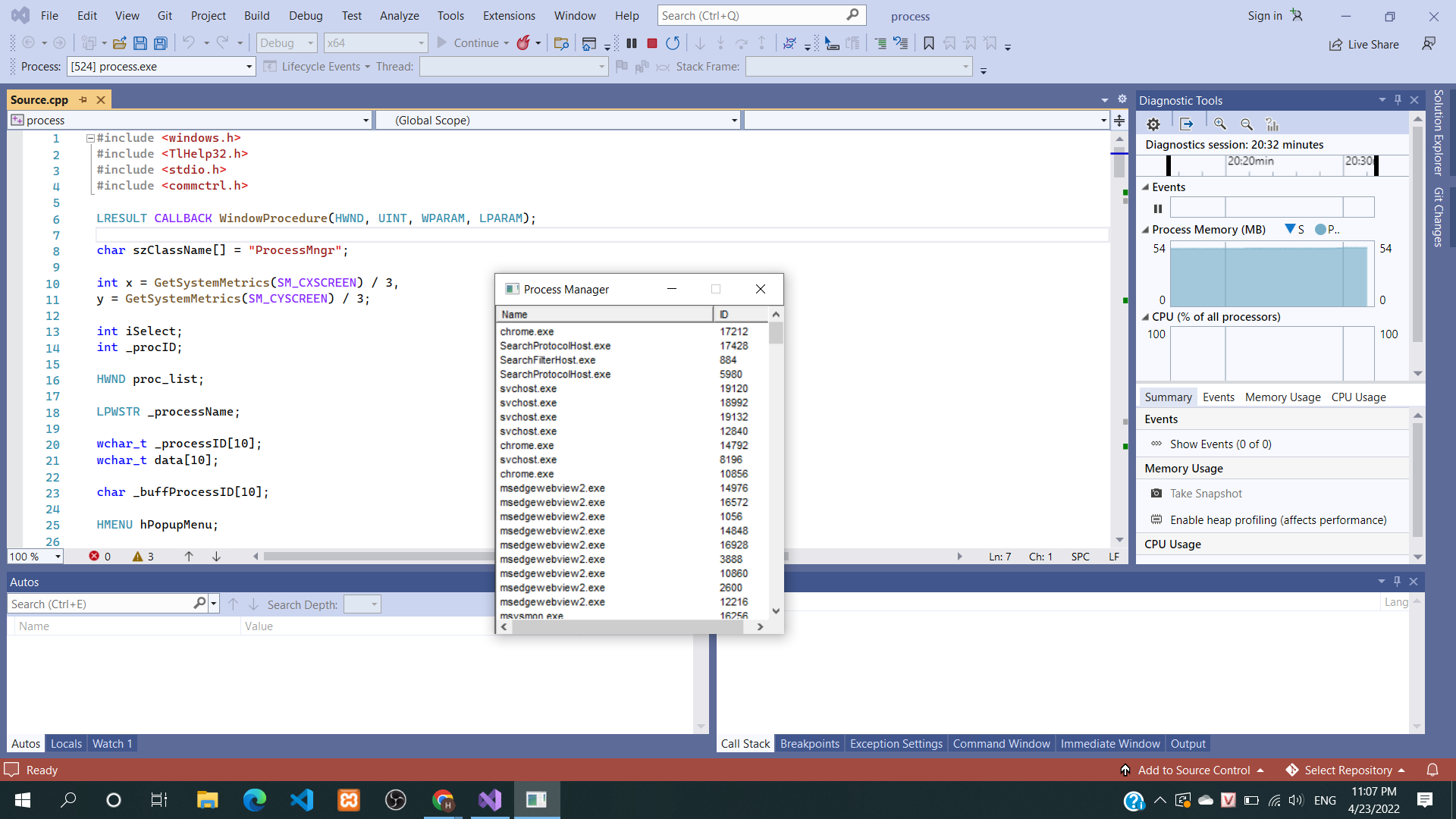
}

### 2.3.2 Hình ảnh màn hình thực nghiệm chương trình

* Hình ảnh chương trình đang chạy task manager

****

* Hình ảnh chương trình sau khi dừng task manager



## 

## 

# CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 3.1 Kết luận

Ứng dụng gọi ngắt quản lý tiến trình là một phần mềm hỗ trợ người dùng trên máy tính hệ điều hành Windows.

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

ứng dụng hỗ trợ gọi ngắt tiến trình cho hệ điều hành windows. Tuy nhiên do kiến thức còn nhiều hạn chế về mặt nội dung nên chương trình không tránh được có nhiều sai sót, nhóm em 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 và thêm một số thông tin khác như CPU, Memory, Disk…Đồng thời áp dụng kiến thức môn học để phát triển ứng dụng hơn nữa hỗ trợ hệ điều hành Windows.

## 

# TÀI LIỆU THAM KHẢO

**[1]**[**https://dhthuyloi.blogspot.com/2014/02/tai-lieu-he-ieu-hanh.html**](https://dhthuyloi.blogspot.com/2014/02/tai-lieu-he-ieu-hanh.html)

**[2]**[**https://docs.microsoft.com/en-us/windows/win32/api/**](https://docs.microsoft.com/en-us/windows/win32/api/)

**[3]**[**https://docs.microsoft.com/en-us/windows/win32/api/commctrl/**](https://docs.microsoft.com/en-us/windows/win32/api/commctrl/)

**[4]**[**https://docs.microsoft.com/en-us/windows/win32/winmsg/using-window-procedures**](https://docs.microsoft.com/en-us/windows/win32/winmsg/using-window-procedures)