Tuần 5: Tìm hiểu về chương trình cửa sổ với MASM

Nội dung:

- The Window Application Entry Point: WinMain
- Window Handler Procedure: WndProc
- Window Messages

Nguồn tham khảo:

Creating a Window - Win32 apps | Microsoft Docs
WinMain The Application Entry Point - Win32 apps | Microsoft Docs
Tutorial: Getting Started (winprog.org)
Windows programs with masm32 · Cogs and Levers (tuttlem.github.io)

1. **The Window Application Entry Point: WinMain**

WinMain là một hàm bắt đầu do người dùng quy định cho một chương trình cửa sổ.
Cấu trúc hàm WinMain (hay wWinMain)

```
C++

int __clrcall WinMain(
  [in] HINSTANCE hInstance,
  [in] HINSTANCE hPrevInstance,
  [in] LPSTR     lpCmdLine,
  [in] int       nShowCmd
);
```

- **hInstance**: xử lý module thực thi chương trình.
- **hPrevInstance**: từng được sử dụng trên phiên bản 16-bit Windows. Trên Win32 luôn có giá trị NULL.
- **lpCmdLine**: là một xâu ký tự Unicode chứa tham số command line (không chứa tên chương trình)
- **nShowCmd**: là một số nguyên truyền cho hàm ShowWindow(), có tác dụng như cờ thông báo chương trình chính được thu nhỏ, phóng to hay hiển thị bình thường.

Tạo cửa sổ chương trình:

- Đăng ký window class:

```
    ; Load default icon
    push IDI_APPLICATION
    push NULL
    call LoadIcon
    mov wc.hIcon, eax
    mov wc.hIconSm, eax

    ; Load default cursor
    push IDC_ARROW
    push NULL
    call LoadCursor
    mov wc.hCursor, eax
call RegisterClassEx
```

wc_cbSize: Kích thước của cấu trúc
wc_Style: Class Styles (CS_*) (phân biệt với Window Styles (WS_*)). Thuộc tính này thường được đặt bằng 0.
wc_lpfnWndProc: Con trỏ trỏ tới window procedure cho window class này.
wc_cbClsExtra: Số byte cần để cấp cho class hiện tại trong bộ nhớ. Thường đặt bằng 0.

wc_cbWndExtra: Số byte cần để cấp cho mỗi window. Thường đặt bằng 0.
wc_hInstance: tương tự như hInstance ở WinMain().
wc_hicon: điều khiển icon lớn (thường là 32x32) của class. Icon này hiện khi người dùng nhấn Alt+Tab. Nếu đặt NULL, hệ thống cung cấp icon mặc định.
wc_hcursor: điều khiển cursor của class. Nếu đặt NULL, hệ thống cung cấp cursor mặc định.
wc_hbrBackground: điều khiển background của class, quy định màu của window.
wc_lpszMenuName: tên của menu dùng cho window của class.
wc_lpszClassName: tên của class
wc_hIconSm: điều khiển icon nhỏ (thường là 16x16) của class. Icon này hiện ở taskbar và trên góc trái trên của window. Nếu đặt NULL, hệ thống cung cấp icon mặc định.

- Tạo Window

```
; after register ClassName, we use it to create windows compond
; https://msdn.microsoft.com/en-us/library/windows/desktop/ms632680(v=vs.85).aspx
push NULL
push hInstance
push NULL
push NULL
push WIN_HEIGHT
push WIN_WIDTH
push CW_USEDEFAULT
push CW_USEDEFAULT
push WS_OVERLAPPEDWINDOW
push offset AppName
push offset ClassName
push WS_EX_CLIENTEDGE
call CreateWindowEx
mov hwnd, eax
```

- Hiển thị Window

```
; display window
; https://msdn.microsoft.com/en-us/library/windows/desktop/ms633548(v=vs.85).aspx
push CmdShow
push hwnd
call ShowWindow

; update window
; https://msdn.microsoft.com/en-us/library/windows/desktop/dd145167(v=vs.85).aspx
push hwnd
call UpdateWindow
```

2. Window Handler Procedure: WndProc

```
126  WndProc proc    hWin    :dword,
127                  uMsg    :dword,
128                  wParam  :dword,
129                  lParam  :dword
130
131          .if uMsg == WM_DESTROY
132
133                  invoke  PostQuitMessage, 0
134
135                  xor     eax, eax
136                  ret
137
138          .endif
139
140          invoke  DefWindowProc, hWin, uMsg, wParam, lParam
141
142          ret
143
144  WndProc endp
```

WndProc là bộ não của chương trình. WndProc nhận các message được gửi đến dưới tên biến uMsg (DWORD).

- **hWin**: điều khiển window.
- **uMsg**: message gửi tới từ WinMain.
  About Message and Message Queues
- **wParam**: thông tin bổ sung cho message. Giá trị phụ thuộc vào giá trị của tham số uMsg.
- **lParam**: thông tin bổ sung cho message. Giá trị phụ thuộc vào giá trị của tham số uMsg.

Ví dụ:

uMsg = WM_DESTROY. Chương trình chính gọi hàm để thoát PostQuitMessage(). Hàm PostQuitMessage() gửi WM_QUIT message đến message queue. Hệ thống sẽ thoát Messages Loop và trả giá trị FALSE cho hàm GetMessage().

3. Window Message

Một chương trình có giao diện cửa sổ cần phải phản hồi các thao tác của người dùng như click chuột, nhập phím, chạm màn hình,… và phản hồi yêu cầu hệ thống. Các event này có mọi lúc trong khi chương trình chạy. Để chương trình có thể đáp ứng các yêu cầu, Windows sử dụng mô hình truyền message. Hệ thống giao tiếp với chương trình thông qua các message. Mỗi message được xác định bởi một giá trị nguyên xác định duy nhất cho từng event.

Ví dụ: Khi người dùng nhấn chuột trái, window nhận message WM_LBUTTONDOWN với giá trị là 0x0201.

Để gửi message tới window, hệ điều hành gọi window procedure đã đăng ký cho window đó.

- **Message Loop**:

```
279     ;Message Loop
280     MESSAGE_LOOP:
281         ; get message
282         ; https://msdn.microsoft.com/en-us/library/windows/desktop/ms644936(v=vs.85).aspx
283         push 0
284         push 0
285         push NULL
286         push offset msg
287         call GetMessage
288
289         ; return in eax
290         ; if the function retrieves a message other than WM_QUIT, the return value is nonzero.
291         ; if the function retrieves the WM_QUIT message, the return value is zero.
292         test eax, eax
293         jle END_LOOP
294
295         ; translate virtual-key messages into character messages - ASCII in WM_CHAR
296         ; https://msdn.microsoft.com/en-us/library/windows/desktop/ms644955(v=vs.85).aspx
297         push offset msg
298         call TranslateMessage
299
300         ; sends the message data to the window procedure responsible for the specific window the message is for.
301         ; https://msdn.microsoft.com/en-us/library/windows/desktop/ms644934(v=vs.85).aspx
302         push offset msg
303         call DispatchMessage
304
305         jmp MESSAGE_LOOP
306
307     END_LOOP:
308         mov eax, msg.wParam
```

Mỗi khi người dùng thực hiện thao tác, hệ thống ghi nhận, sinh ra các message và gửi các message tới message queue của chương trình. Bằng việc gọi hàm GetMessage(), chương trình xóa message hiện thời trên queue và dùng nó cho tiến trình. Khi có message lấy từ queue, chương trình thực hiện TranslateMessage() đổi các message virtual-key sang thành message char. Cuối cùng DispatchMessage() gửi message char này ra ngoài window đích (có thể là main window của chương trình).