CƠ BẢN VỀ LẬP TRÌNH WIN32 API

1

Lập trình Windows cơ bản

- Application Programming Interface: Giao diện lập trình ứng dụng
- Cung cấp thư viện liên kết động (.dll) chứa các hàm truy cập tài nguyên trong hệ thống (Windows)
- Các ứng dụng có thể truy cập đến các hàm API
- Kiểu dữ liệu HANDLE: số nguyên 32-bit được HĐH sử dụng để định danh cho một đối tượng tài nguyên nào đó (file, cửa sổ, socket,…)
  - Ví dụ: HWND, HINSTANCE, SOCKET

2

Một số kiểu dữ liệu

Kiểu	Ý nghĩa
BYTE	Số nguyên 8-bit không dấu
WORD	Số nguyên 16-bit không dấu
DWORD	Số nguyên 32-bit không dấu
UINT	Số nguyên 32-bit không dấu
LONG	Số nguyên 32-bit có dấu
LRESULT	Số nguyên 32-bit có dấu trả về bởi hàm xử lý thông điệp
LPSTR	Con trỏ xâu ký tự kiểu ANSI (8-bit)
LPWSTR	Con trỏ xâu ký tự kiểu Unicode(16 bit)
WPARAM	Số nguyên không dấu (32-bit với x86 và 64-bit với x64)
LPARAM	Số nguyên có dấu (32-bit với x86 và 64-bit với x64)
LPVOID	Con trỏ kiểu void
ATOM	Số nguyên 16-bit không dấu
LPTSTR	Con trỏ xâu ký tự kiểu tự thích nghi(8-bit hoặc 16-bit)

3

Quy ước lời gọi hàm của C/C++

- Mô tả cách thức hệ thống xử lý lời gọi hàm
  - Thứ tự truyền tham số
  - Cách thức truyền tham số
  - Thanh ghi dành riêng
  - Vào ra trên stack
- __stdcall
  - Truyền từ phải qua trái
  - Kiểu truyền mặc định: tham trị
  - Hàm được gọi (callee) lấy giá trị tham số từ đỉnh stack
  - Hàm được gọi xóa stack trước khi trả về kết quả
  - Tên hàm: bắt đầu bởi ký tự '_'
- __cdecl: tương tự __stdcall
  - Hàm gọi xóa stack sau khi hàm được gọi trả về kết quả

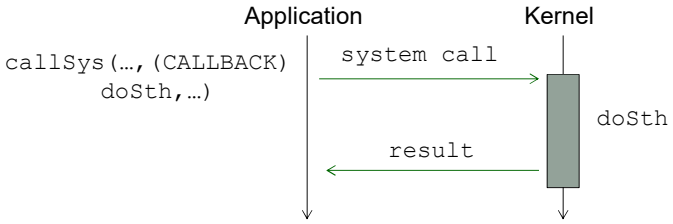
4

Một số quy ước gọi hàm của Windows

- Hàm kiểu CALLBACK

```
#define CALLBACK __stdcall
```

  - Hàm callback: Hàm mà địa chỉ của nó là tham số được truyền trong lời gọi một hàm khác
  - Truyền hàm callback khi thực hiện lời gọi hệ thống: khai báo với quy ước gọi hàm là CALLBACK hoặc tương đương



- Hàm kiểu WINAPI: sử dụng trong ứng dụng Windows

```
#define WINAPI __stdcall
```

5

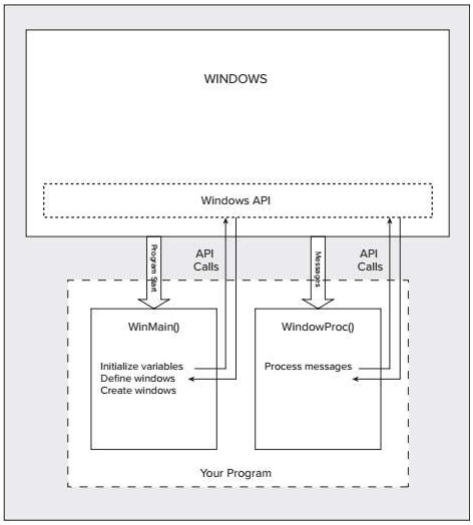
Lập trình hướng sự kiện

- Là mô hình lập trình mà luồng thực thi chương trình được quyết định bởi các sự kiện xảy ra
- Ví dụ: Sự kiện trên cửa sổ
  - Hệ thống (Windows) ghi lại mỗi sự kiện xảy ra (event) trong một thông điệp (message) và đặt trong hàng đợi thông điệp (messages queue)
  - HĐH Windows đảm nhiệm việc truyền thông điệp vào cửa sổ của ứng dụng đó
  - Hàm kiểu CALLBACK cần được định nghĩa để xử lý các thông điệp mà cửa sổ nhận được
    - Với những thông điệp không xử lý, có thể truyền lại cho hệ thống qua hàm DefWindowProc()

6

Cấu trúc chương trình

- Hàm WINAPI WinMain():
  - Khởi tạo giá trị biến
  - Định nghĩa cửa sổ
  - Khởi tạo cửa sổ
- Hàm CALLBACK WindowProc():
  - Xử lý thông điệp



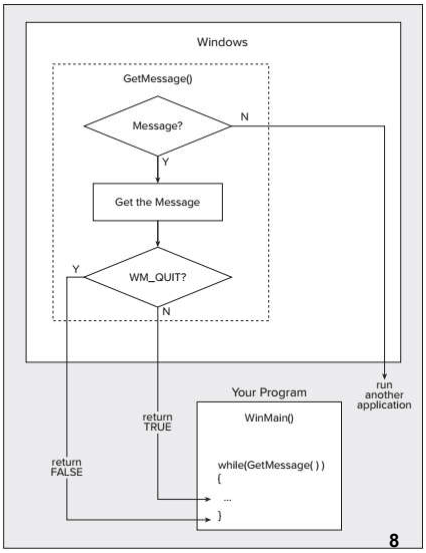
7

Hàm WinMain()

- Xử lý hàng đợi thông điệp

```
MSG msg;
while (GetMessage(&msg,NULL,0,0)){
   //Dịch sự kiện thành thông điệp
   TranslateMessage (&msg);
   //Lấy thông điệp
   DispatchMessage(&msg);
}
```

- Hàm GetMessage() trả về 0 khi thông điệp chứa định danh WM_QUIT báo chương trình kết thúc



8

Cấu trúc MSG

```
typedef struct tagMSG {
   HWND   hwnd;      // Cửa sổ nhận thông điệp
   UINT   message;   // Định danh thông điệp
   WPARAM wParam;    // Nội dung thông điệp
   LPARAM lParam;    // Nội dung thông điệp
   DWORD  time;      // Thời điểm thông điệp được đưa lên
   POINT  pt;        // Vị trí con trỏ khi thông điệp được
                     // đưa lên
} MSG, *PMSG, *LPMSG;
```

9

Lớp cửa sổ WNDCLASS

```
typedef struct tagWNDCLASS {
   UINT      style;
   WNDPROC   lpfnWndProc;
   int       cbClsExtra;
   int       cbWndExtra;
   HINSTANCE hInstance;
   HICON     hIcon;
   HCURSOR   hCursor;
   HBRUSH    hbrBackground;
   LPCTSTR   lpszMenuName;
   LPCTSTR   lpszClassName;
} WNDCLASS, *PWNDCLASS;
```

- Lớp WNDCLASSEX bổ sung thêm trường **cbSize** chứa kích thước cấu trúc và **hIconSm** chứa icon nhỏ của cửa sổ

10

Cấu trúc cửa sổ WNDCLASS

Tên trường	Ý nghĩa	Gán giá trị
`style`	Kiểu hiển thị	
`lpfnWndProc`	Con trỏ hàm xử lý cửa sổ	
`cbClsExtra`	Kích thước (byte) cấp phát thêm sau biến cấu trúc WNDCLASS	0: gán mặc định bởi hệ thống
`cbWndExtra`	Kích thước (byte) cấp phát thêm cho tiến trình	0: gán mặc định bởi hệ thống
`hInstance`	Định danh của tiến trình	
`hIcon`	Định danh của icon	Gọi hàm `LoadIcon()`
`hCursor`	Định danh con trỏ chuột	Gọi hàm `Cursor()`
`hbrBackground`	Định danh chổi vẽ	
`lpszMenuName`	Tên menu	
`lpszClassName`	Tên cửa sổ	

11

Đăng ký và tạo cửa sổ

- Đăng ký cửa sổ

```
ATOM RegisterClass(CONST WNDCLASS * lpWndClass);
ATOM RegisterClassEx(CONST WNDCLASS * lpWndClass);
```

- Tạo cửa sổ

```
HWND WINAPI CreateWindow(
   LPCTSTR   lpClassName,     // Tên cửa sổ
   LPCTSTR   lpWindowName,    // Tên hiển thị
   DWORD     dwStyle,         // Kiểu cửa sổ
   int       x,               // Hoành độ vị trí hiển thị
   int       y,               // Tung độ vị trí hiển thị
   int       nWidth,          // Kích thước ngang
   int       nHeight,         // Kích thước cao
   HWND      hWndParent,      // Cửa sổ cha
   HMENU     hMenu,           // Định danh menu
   HINSTANCE hInstance,       // Định danh tiến trình
   LPVOID    lpParam          // Con trỏ tham số truyền
                              // cho cửa sổ
);
```

12

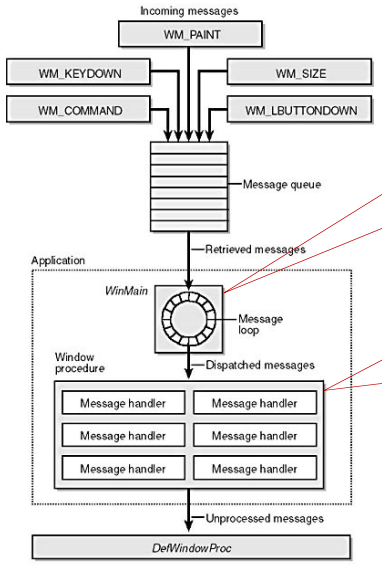
Khai báo hàm CALLBACK

```
LRESULT CALLBACK windowProc(
   HWND hWnd,          // Định danh cửa sổ
   UINT message,       // Định danh thông điệp
   WPARAM wParam,      // Nội dung thông điệp
   LPARAM lParam){     // Nội dung thông điệp

   switch(message){
        case WM_XXX:
        …
        case WM_DESTROY:       // Cửa sổ bị hủy
           PostQuitMessage(0);
           …
        case WM_CLOSE:         // Cửa sổ bị đóng
           DestroyWindow(hWnd);
   }
   DefWindowProc(hWnd, message, wParam, lParam);
}
```

13

Hoạt động của chương trình




```
int WINAPI WinMain (
   HINSTANCE hInstance,
   HINSTANCE hPrevInstance,
   LPSTR lpCmdLine,
   int nCmdShow)
{
}
```

```
LRESULT CALLBACK WinProc (
        HWND hWnd,
        UINT message,
        WPARAM wParam,
        LPARAM lParam)
{
}
```

14