Computer Programming
Windows API 26th Lecture

엄현상 (Eom, Hyeonsang)
컴퓨터공학부
서울대학교

©Copyrights 2010 Eom, Hyeonsang All Rights Reserved
순서

- Introduction to Windows
- What is Windows API?
- Application Development Environments
- Message Driven System
- Other Backgrounds
- Function WinMain
- Function WndProc
- API Example : Hello, World!
- Possible Topics for Final
Introduction to Windows

History of Windows
- 85.11 Windows 1.0
- 87.11 Windows 2.0
- 92.4 Windows 3.1
- 93.7 Windows NT 3.1
- 95.8 Windows 95
- 96.8 Windows NT 4.0
- 98.6 Windows 98
- 00.2 Windows 2000
- 02.9 Windows XP
- 07.1 Windows Vista
- 09.7 Windows 7
Introduction to Windows Cont’d

- **Merits**
  - Being the Most Widely Used in Personal Computing
  - GUI: Graphical User Interface
  - Multitasking
  - Standard User Interface
    - User Adapts to New Program Quickly

- **Demerits**
  - Very Commercial Product!
    - Not Open Source!
[참고] X Windows

: Standard Graphical Engine for Unix/Linux

Difference with Microsoft Windows

- Platform Independent
- Network Transparent

- X Protocol
  - Provide a client-server architecture at the application level
  - Separate the processing and display for an application
    - X Server: Low-level interface for controlling screen
    - X Client: Processing part of the application
What is Windows API?

- API (Application Programming Interface)
- Possibly Making Windows System Calls
Application Development Environments

Windows Software Development Kit (SDK)

- Providing API Libraries, Documents, Tools, etc.

  - Using API Directly
    - High Performance, Fine-Grain Control
    - Low Productivity

Class Libraries

  - Wrapping APIs to Classes
    - Convenient and Powerful
    - E.g., MFC (Microsoft Foundation Class) for Visual C++

Rapid Application Development (RAD) Tools

  - Providing Users—Don’t-Care Visual Tools
    - High Productivity, but Low Performance
    - E.g., Visual Basic, Delphi, Power Builder, etc.
Message Driven System

- Content of a Message
  - Event Information
    - Change in the System
    - Change between the User and System
    - (User) Action in a Window

- Role of OS Regarding a Message
  - Inspecting What Events Have Occurred, and Generating the Corresponding Message
  - Enqueuing the Message into a Message Queue Belonging to Each Program
Message Driven System Cont’d

- **Use of a Message Queue**
  - Each Program Has a Message Queue
  - Function `WndProc` Processes a Message

[Diagram showing the process of an action, message generation, and enqueue into the message queue.]
Message Driven System Cont’d

- Use of the Message Loop
  - Location
    - End of Function WinMain
  - Functionality
    - Dequeuing from the Message Queue
    - Translating If Necessary
    - Delivering to the Event Handler
Message Driven System Cont’d

```c
int APIENTRY WinMain(...) {
    ...
    while (GetMessage(&message, 0, 0, 0)) {
        TranslateMessage(&message);
        DispatchMessage(&message);
    }
    ...
}
```

- **Return false** if getting a WM_QUIT message, else return true.

- **Translate a keyboard input message**
  - e.g., WM_KEYDOWN & WM_KEYUP
  - → WM_CHAR

- **Deliver the message to function WndProc that specifies what to do**
Message Driven System Cont’d

OS

Application

while(GetMessage(&message, 0, 0, 0)){
    TranslateMessage(&message);
    DispatchMessage(&message);
}

switch(message){
    case WM_CHAR:
        ...
        break;
    case ...
    default: return DefWindowProc();
}
Other Backgrounds

- Handles
  - Variables Distinguishable from Other Resources
  - 32 bit Integer Type Value
  - Those Made by OS, Not the Users
    - Similarly, "fd=open()" in Linux
  - Types: HWND, HPEN, HBRUSH, HDC, etc.
    - Represent a Handle of Each Resource
Other Backgrounds Cont’d

- Way for Printing

Win32 API

Application

G D I

Device Driver

Output Device

Graphic Device Interface
Other Backgrounds Cont’d

Printing Procedure

Get a device context handle for printing on screen

```c
HDC hDC = GetDC(hWnd);
```

Print out using the handle via GDI

```c
TextOut(hDC, 0, 0, ”Hello” , 5);
LineTo(hDC, 100, 100);
```

Release the device context handle

```c
ReleaseDC(hWnd, hDC);
```
Function WinMain

Entry Point of Program

- Similar to “int main(int, char**);”

```c
int APIENTRY WinMain(HINSTANCE hinstance, HINSTANCE hPrevInstance, LPSTR lpszCmdParam, int nCmdShow);
```

<table>
<thead>
<tr>
<th>hInstance</th>
<th>Instance handle of window</th>
</tr>
</thead>
<tbody>
<tr>
<td>hPrevInstance</td>
<td>Previous instance handle</td>
</tr>
<tr>
<td></td>
<td>Currently this is always NULL</td>
</tr>
<tr>
<td>lpszCmdParam</td>
<td>Command arguments (like argv)</td>
</tr>
<tr>
<td>nCmdShow</td>
<td>Shape of program</td>
</tr>
<tr>
<td></td>
<td>(minimize, normal, maximize, ...)</td>
</tr>
</tbody>
</table>
Function WinMain Cont'd

1. start
2. create & register a window class
3. create a window
4. show the window
5. get a message
6. WM_QUIT
   - dispatch a message
   - translate a message
5a. true
   - end
5b. false
Function WndProc

Window Procedure
- Specify What to Do When an Event Occurs

LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
                         WPARAM wParam, LPARAM lParam);

<table>
<thead>
<tr>
<th>hWnd</th>
<th>Instance handle of window</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Types of message</td>
</tr>
<tr>
<td>wParam</td>
<td>Additional information of message (different according to message)</td>
</tr>
<tr>
<td>lParam</td>
<td></td>
</tr>
</tbody>
</table>
API Example: Hello, World!

- Using MS Visual C++ 6.0
API Example: Hello, World! Cont’d

- Make a Workspace for the Example
  - Create a Project
    - Select menu: File → New → Projects

[Image of a workspace creation interface with highlighted options: select “Win32 Application” and specify the project name.]
API Example: Hello, World! Cont’d

□ Select a Default Skeleton Code

- empty code
- a simple program: having empty functions (WinMain, WndProc, …)
- a simple program: displaying “Hello World!” and having a simple menu
API Example: Hello, World! Cont’d

- Add a New Source File into the Project
  - Select menu: File → New → Files

  ![Image of file selection](image.png)

  - Specify the source file name
  - Select "C++ Source File"

API Example: Hello, World! Cont’d

Source Code

#include <windows.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

HINSTANCE g_hInst;

LPSTR lpszClass="Hello, World!";

int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance ,LPSTR lpszCmdParam,int nCmdShow)
{
    HWND hWnd;
    MSG Message;
    WNDCLASS WndClass;
    g_hInst=hInstance;

    program title

API Example: Hello, World! Cont’d

```c
WndClass.cbClsExtra=0;
WndClass.cbWndExtra=0;
WndClass.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);
WndClass.hCursor=LoadCursor(NULL,IDC_ARROW);
WndClass.hIcon=LoadIcon(NULL,IDI_APPLICATION);
WndClass.hInstance=hInstance;
WndClass.lpfnWndProc=(WNDPROC)WndProc;
WndClass.lpszClassName=lpszClass;
WndClass.lpszMenuName=NULL;
WndClass.style=CS_HREDRAW | CS_VREDRAW;
RegisterClass(&WndClass);

hWnd=CreateWindow(lpszClass,lpszClass,WS_OVERLAPPEDWINDOW,
                  CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
                  CW_USEDEFAULT,NULL,(HMENU)NULL,hInstance,NULL);
ShowWindow(hWnd,nCmdShow);
```

API Example: Hello, World! Cont’d

```c
while(GetMessage(&Message, 0, 0, 0)) {
    TranslateMessage(&Message);
    DispatchMessage(&Message);
}
return Message.wParam;
```
API Example: Hello, World! Cont'd

```c
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
    WPARAM wParam, LPARAM lParam) {

    switch(message) {
    case WM_DESTROY: PostQuitMessage(0); break;
    case WM_PAINT:
        HDC hDC; PAINTSTRUCT ps;
        hDC = BeginPaint(hWnd, &ps);
        TextOut(hDC, 10, 10, "Hello, World!", 14);
        EndPaint(hWnd, &ps); break;
    default: return DefWindowProc(hWnd, message, wParam, lParam);
    }

    return 0;
}
```

- when a window needs repaint
- send WM_QUIT to itself
- pass the message to the default window procedure by default
API Example: Hello, World! Cont’d

Compilation

- Ctrl+F7: Compile a Source File
- F7: Build a Executable File
- Ctrl+F5: Build and Execute the Program without Debugging
- F5: Build and Execute the Program with Debugging
Possible Topics for Final

- **Unix/Linux**
  - C Program Memory Layout
  - File System
  - Interprocess Communication

- **Utilities and Editors**
  - Emacs

- **Batch and Shell Programming**

- **C Compiler and Linker**
  - Static vs Shared Library

- **Gdb & Make**

- **Modularity and Abstraction in C**
  - Software Modular Design
Possible Topics for Final Cont’d

- Scoping
- C Pointers
  - Call by Reference
- Memory Management in C
  - Possible Errors in Dynamic Memory Allocation
- Libraries
  - Standard I/O Library
- Object-Oriented Programming
  - Design Principles
- C, C++, or Java Programming
  - Static