ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

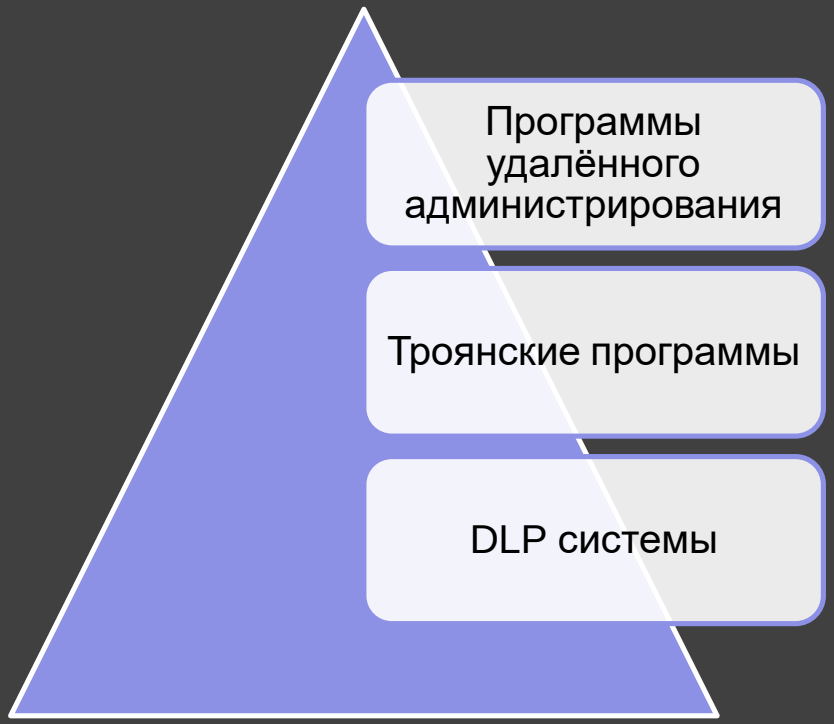
Ставьте + если все хорошо

1

Сообщения Windows

Что такое HOOK?

HOOK – это механизм перехвата сообщений



Сообщения Windows

Взаимодействие между

Графическими

элементами

Осуществляется с

помощью сообщений

```
int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
                      _In_opt_ HINSTANCE hPrevInstance,
                      _In_ LPWSTR    lpCmdLine,
                      _In_ int       nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // TODO: Place code here.

    // Initialize global strings
    LoadStringW(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadStringW(hInstance, IDC_WIN32PROJECT1, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Perform application initialization:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }

    HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_WIN32PROJECT1));

    MSG msg;

    // Main message loop:
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }

    return (int) msg.wParam;
}
```

Сообщения

OS Windows генерирует сообщения на:

- каждое событие ввода
- изменения в системе
- Кастомные события

Сообщения отправляются оконной процедуре графического приложения. Программа может создать и отправить сообщение сама себе, либо другой программе

Сообщения Windows

OTUS

Сообщение представляет из себя:

Id самого сообщения

- описывает что именно произошло. Клавиатурное событие, оконное событие и тд

Два дополнительных параметра

- wParam и lParam, которые дополняют событие информацией

Цикл обработки сообщений

```
// Main message loop:
while (GetMessage(&msg, nullptr, 0, 0))
{
    if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
```

GetMessage извлекает сообщение из очереди сообщений. *Если в очереди пусто, то поток блокируется. Второй параметр – дескриптор окна, от которого принимается сообщение. Третий и четвёртые параметры – диапазон id принимаемых сообщений*

TranslateMessage нужна, чтобы сконвертировать сообщение нажатой виртуальной клавиши (например, WM_KEYDOWN) в другое сообщение - символьное WM_CHAR. Его параметр будет содержать символ нажатой клавиши Результат помещает в очередь сообщений

DispatchMessage извлечённое сообщение отправляет в очередь на обработку оконной процедуре

Сообщения Windows

Каждое окно имеет свою

оконную процедуру.

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_COMMAND:
        {
            int wmId = LOWORD(wParam);
            // Parse the menu selections:
            switch (wmId)
            {
            case IDM_ABOUT:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                break;
            case IDM_EXIT:
                DestroyWindow(hWnd);
                break;
            default:
                return DefWindowProc(hWnd, message, wParam, lParam);
            }
        }
        break;
    case WM_PAINT:
        {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hWnd, &ps);
            // TODO: Add any drawing code that uses hdc here...
            EndPaint(hWnd, &ps);
        }
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

WindowProc

OTUS

c++ Copy

```
LRESULT CALLBACK WindowProc(
  _In_ HWND   hwnd,
  _In_ UINT   uMsg,
  _In_ WPARAM wParam,
  _In_ LPARAM lParam
);
```

Hwnd – handle окна, которому предназначено сообщение

uMsg – тип сообщения

wParam/lParam – параметры, которые могут иметь разный смысл, в зависимости, от uMsg

Пример

OTUS

“Ввести” текст в окно notepad

```
int main()
{
    HWND hWndCalc, hEditWnd;

    hWndCalc = FindWindowA("Notepad", NULL);
    if (hWndCalc)
    {
        hEditWnd = FindWindowEx(hWndCalc, NULL, L"Edit", NULL);
        if (hEditWnd)
            SendMessage(hEditWnd, WM_SETTEXT, 0, (LPARAM)L"TEST");
    }
    return 0;
}
```

Пример

FindWindow – ищет окно по его классу

FindWindowEx – ищет дочернее окно

SendMessage – отправляет сообщение

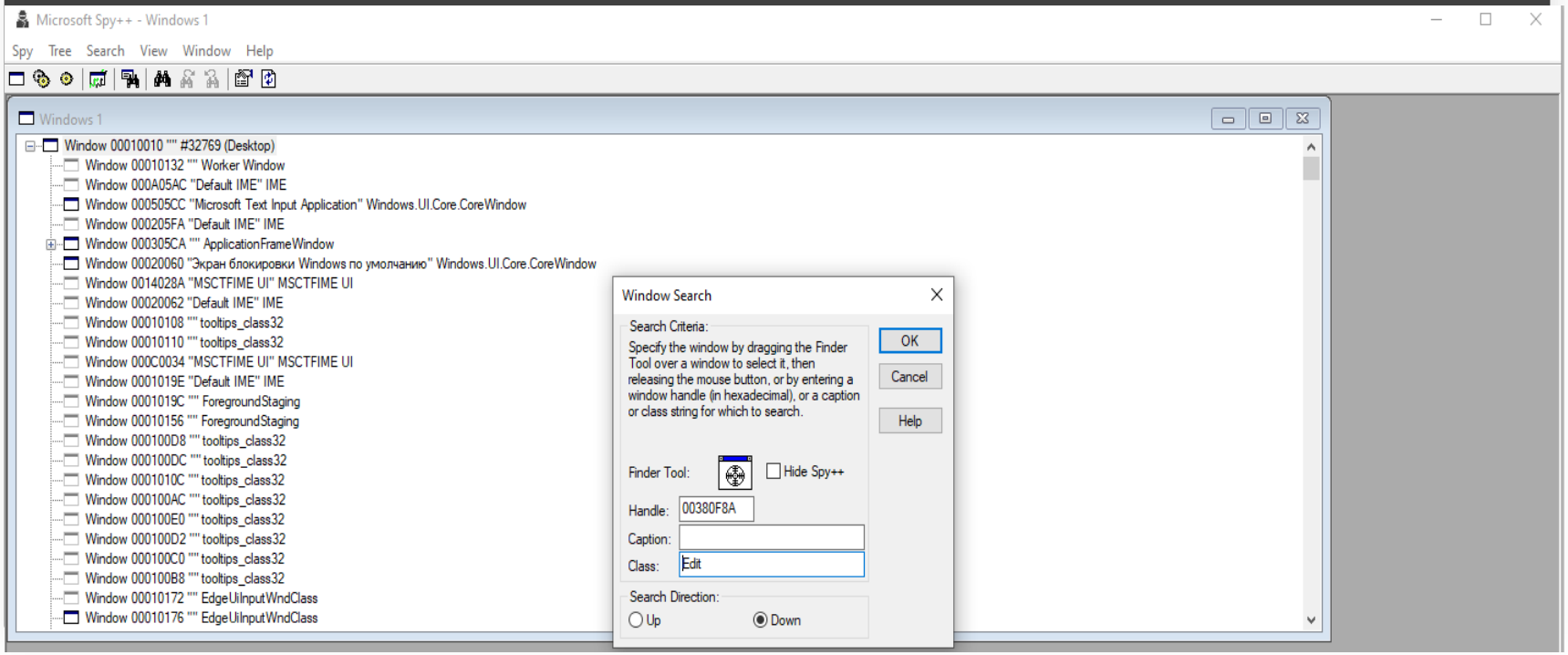
```
int main()
{
    HWND hWndCalc, hEditWnd;

    hWndCalc = FindWindowA("Notepad", NULL);
    if (hWndCalc)
    {
        hEditWnd = FindWindowEx(hWndCalc, NULL, L"Edit", NULL);
        if (hEditWnd)
            SendMessage(hEditWnd, WM_SETTEXT, 0, (LPARAM)L"TEST");
    }
    return 0;
}
```

Утилита spyxx

Для определения классов окон, можно использовать утилиту spyxx.exe, которая входит в комплект VS

```
D:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tools
```

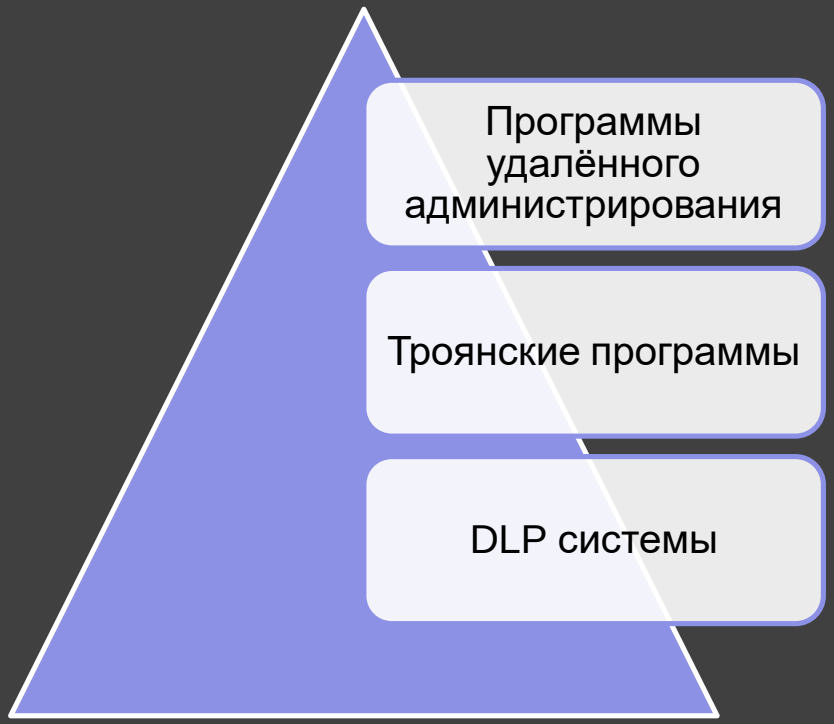


2

Ловушки Windows

Что такое HOOK?

HOOK – это механизм перехвата сообщений



Что такое HOOK?

Хуки бывают глобальные и локальные (на определённый поток)

Установить хук можно вызовом функции **SetWindowsHookExA**

C++ Copy

```
HHOOK SetWindowsHookExA(
  int       idHook,
  HOOKPROC  lpfn,
  HINSTANCE hmod,
  DWORD     dwThreadId
);
```

SetWindowsHookExA

C++ Copy

```
HHOOK SetWindowsHookExA(
  int       idHook,
  HOOKPROC  lpfn,
  HINSTANCE hmod,
  DWORD     dwThreadId
);
```

idHook - тип устанавливаемого перехвата

Lpfn - Наш обработчик хука

hmod - базовый адрес Решника содержищего обработчик хука

dwThreadId - id потока, куда вставляется хук (0 - если глобальный)

idHook

```
WH_MOUSE - считывается событие мыши

WH_KEYBOARD - считывается событие WM_KEYDOWN или WM_KEYUP

WH_CBT - вызывается перед обработкой оконных сообщений
```

Очередь ловушек

При установке двух одинаковых ловушек, они выстраиваются в очередь. Для передачи выполнения кода следующей ловушке, используется CallNextHookEx

C++ Copy

```
LRESULT CallNextHookEx(
  HHOOK  hhk,
  int    nCode,
  WPARAM wParam,
  LPARAM lParam
);
```

Глобализатор

Один из способов подгрузки своей DLL в АП сторонних процессов – использование хуков

```
HHOOK hook = NULL;

LRESULT CALLBACK GetMsgProc(int nCode, WPARAM wParam, LPARAM lParam)
{
    return CallNextHookEx(hook, nCode, wParam, lParam);
}

void APIENTRY SetHook(HMODULE hInstance, BOOL setHook = TRUE)
{
    hook = SetWindowsHookEx(WH_GETMESSAGE, GetMsgProc, hInstance, 0);
    if (hook)
    {
        MessageBoxA(GetForegroundWindow(), "Hooked!", "", 0);
    }
}

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        SetHook(hModule);
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

OTUS

Вопросы???

Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!