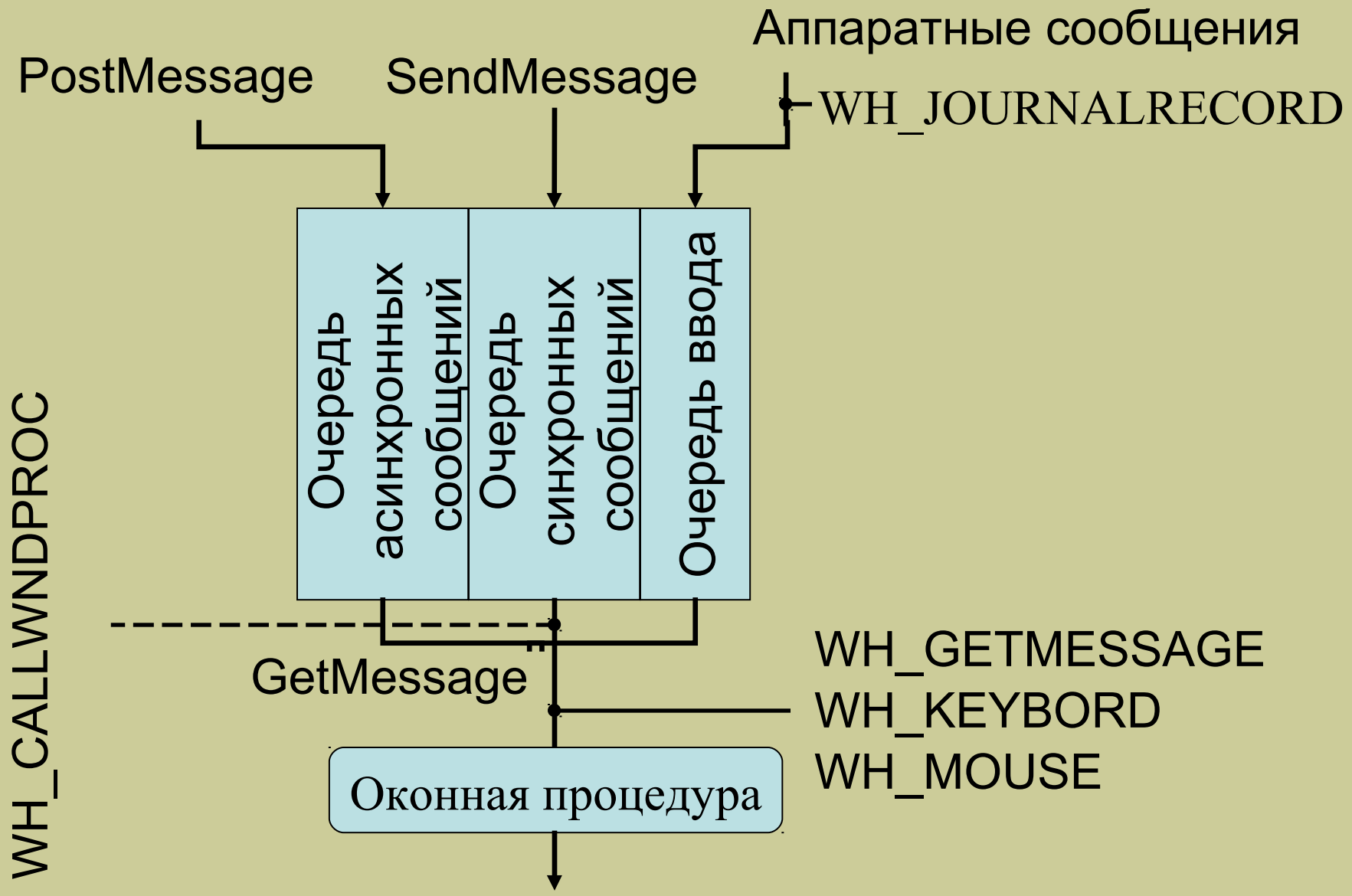
Лекция 17

Ловушки Windows

- Типы ловушек.
- Процедуры ловушек.
- Установка ловушек.

Типы ловушек



WH_CALLWNDPROC – система вызывает процедуру ловушки **CallWndProc** до передачи сообщения оконной процедуре адресату. Этой ловушке разрешается анализировать сообщения, но не изменять их.

WH_GETMESAGE – система вызывает процедуру ловушки **GetMsgProc** до передачи сообщения оконной процедуре адресату. Этой ловушке *разрешается изменять* сообщения.

WH_KEYBOARD – система вызывает процедуру ловушки **KeyboardProc** когда извлеченное из очереди сообщение исходит от клавиатуры (WM_KEYUP или WM_KEYDOWN).

WH_MOUSE – система вызывает процедуру ловушки **MouseProc** когда извлеченное из очереди сообщение исходит от мыши (например WM_LBUTTONDOWN).

WH_CBT – система вызывает процедуру ловушки **CBTProc** перед созданием, активацией и модификацией окон.

Процедуры ловушек имеют сходную сигнатуру:

```
LRESULT CALLBACK MouseProc(
int nCode, // код  ловушки (напр. HC_ACTION)
WPARAM wParam, // идентификатор сообщения
LPARAM lParam     // указатель на структуру
                  // MOUSEHOOKSTRUCT
);
```

Локальные ловушки.

Глобальные ловушки. **Загружаются в адресное пространства каждого процесса**.

hh.c

```
#include <windows.h>
#include "resource.h«

__declspec(dllimport) void SetFarHook(void);
__declspec(dllimport) void UninstallFarHook(void);

LRESULT CALLBACK DlgProc(HWND hDlg, UINT message,
                  WPARAM wParam, LPARAM lParam);

int APIENTRY WinMain(HINSTANCE hInstance,
                                HINSTANCE hPrevInstance,
                                LPSTR     lpCmdLine,
                                int       nCmdShow)
{
   MSG msg;
    DialogBox(hInstance,(LPCTSTR)IDD_DLGTEST,NULL,
                           (DLGPROC)DlgProc);
```

```
while (GetMessage(&msg, NULL, 0, 0))
      {
         TranslateMessage(&msg);
         DispatchMessage(&msg);
      }
      return msg.wParam;
}

LRESULT CALLBACK DlgProc(HWND hDlg, UINT message,
                  WPARAM wParam, LPARAM lParam){
       char strText[100];
        switch (message){
```

```
case WM_INITDIALOG:
    RegisterHotKey(hDlg,0xB001, MOD_CONTROL |
                                        MOD_ALT, 'W');
     return TRUE;
case WM_HOTKEY:
     ShowWindow(hDlg,SW_SHOW);
     break;
case WM_COMMAND:
      switch (LOWORD(wParam) ){
           case IDOK:
                PostQuitMessage(0);
                return TRUE;
           case IDC_BTN1:
              SetFarHook();
              ShowWindow(hDlg,SW_HIDE);
              break;
```

```
        case IDC_BTN2:
            UninstallFarHook();
            break;
        }
        break;
        default:
         return FALSE;
    }
}
```

resource.h

```
#define IDD_DLGTEST                 101
#define IDC_BTN1                   1001
#define IDC_BTN2                   1002
```

hh.rc

```
#include <windows.h>
#include "resource.h"
IDD_DLGTEST DIALOG DISCARDABLE  0, 0, 80, 80
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Hang the hook",IDC_BTN1,7,7,60,15
    PUSHBUTTON      "Pull off",IDC_BTN2,7,27,60,15
    DEFPUSHBUTTON   "OK",IDOK,7,47,60,15
END
```

Компилятор ресурсов:

```
> rc  hh.rc
```

h1.c

```
#include <windows.h>
HINSTANCE hinstDLL;
HHOOK g_Hook;

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                         )
{
  hinstDLL=(HINSTANCE)hModule;
  return TRUE;
}

LRESULT CALLBACK CBTProc(int nCode, WPARAM
wParam, LPARAM lParam)
{
```

```
CBT_CREATEWND* cbt_crwnd;
    if (nCode < 0)
    return CallNextHookEx(g_Hook, nCode, wParam,
                                                    lParam);
 cbt_crwnd=(CBT_CREATEWND*)lParam;

  switch (nCode) {
   case HCBT_CREATEWND: //код ловушки
if (CompareString(LOCALE_SYSTEM_DEFAULT,
                NORM_IGNORECASE,
                cbt_crwnd->lpcs->lpszName,
                6,
                "Change",
                6)==CSTR_EQUAL)
   DestroyWindow(cbt_crwnd->lpcs->hwndParent);
break;
```

```
default:
        break;
  }
  return CallNextHookEx(g_Hook, nCode, wParam, lParam);
}

__declspec(dllexport) void SetFarHook(void){
g_Hook=SetWindowsHookEx(WH_CBT,CBTProc,hinstDLL,0);
return;
}

__declspec(dllexport) void UninstallFarHook(void){
UnhookWindowsHookEx(g_Hook);
return;
}
```

```
>cl /c h1.c

> link /DLL h1.obj user32.lib
```

```
>cl  hh.c hh.res h1.lib user32.lib
```

h2.c

```
#include <windows.h>
HINSTANCE hinstDLL;
HHOOK g_Hook;

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved){

  hinstDLL=(HINSTANCE)hModule;
  return TRUE;
}
```

```
LRESULT CALLBACK MouseProc(int nCode, WPARAM
                                        wParam, LPARAM lParam) {

  MOUSEHOOKSTRUCT ms;
  char str[64]={'\0'};
  int x,y;

  if (nCode < 0)
      return CallNextHookEx(g_Hook, nCode, wParam, lParam);

if(nCode==HC_ACTION)
   switch (wParam) {
       case WM_RBUTTONDBLCLK :
          sprintf(str,"%i  %i",
                    ((MOUSEHOOKSTRUCT*)lParam)->pt.x,
                     ((MOUSEHOOKSTRUCT*)lParam)->pt.y);
```

```
  MessageBox(NULL,str, "",MB_OK);
  break;
  }
  return CallNextHookEx(g_Hook, nCode, wParam, lParam);
}

__declspec(dllexport) void SetFarHook(void){
        g_Hook=SetWindowsHookEx(WH_MOUSE,
                                        MouseProc,hinstDLL,0);
        return;
}

__declspec(dllexport) void UninstallFarHook(void){
        UnhookWindowsHookEx(g_Hook);
        return;
}
```

Упражнение 1: протестировать разобранные программы

Упражнение 2: проверить адресное пространство текущих процессов после установки глобальной ловушки и после ее снятия, установить наличие или отсутствие модуля ловушки (воспользоваться программами предыдущих лабораторных).

Упражнение 3: записать все события мыши (или клавиатуры) происходящие в системе (используйте, например, ловушки мыши и клавиатуры, а также WM_COPYDATA для передачи данных между процессами).