

Хуки

- В операционной системе Windows хуком называется механизм перехвата особой функцией событий (таких как сообщения, ввод с мыши или клавиатуры) до того, как они дойдут до приложения. Эта функция может затем реагировать на события и, в некоторых случаях, изменять или отменять их.
- Функции, получающие уведомления о событиях, называются *фильтрующими функциями* и различаются по типам перехватываемых ими событий.
- Пример - фильтрующая функция для перехвата всех событий мыши или клавиатуры. Чтобы Windows смогла вызывать функцию-фильтр, эта функция должна быть установлена - то есть, прикреплена - к хуку (например, к клавиатурному хуку).
- Прикрепление одной или нескольких фильтрующих функций к какому-нибудь хуку называется *установкой хука*.

- Если к одному хуку прикреплено несколько фильтрующих функций, Windows реализует очередь функций, причем функция, прикрепленная последней, оказывается в начале очереди, а самая первая функция - в ее конце.
- Когда к хуку прикреплена одна или более функций-фильтров и происходит событие, приводящее к срабатыванию хука, Windows вызывает первую функцию из очереди функций-фильтров.
- Это действие называется вызовом хука. К примеру, если к хуку CBT прикреплена функция и происходит событие, после которого срабатывает хук (допустим, идет создание окна), Windows вызывает CBT-хук, то есть первую функцию из его очереди.
- Для установки и доступа к фильтрующим функциям приложения используют функции **SetWindowsHookEx** и **UnhookWindowsHookEx**.

Приложения могут использовать хуки в следующих целях:

- Обработать или изменить все сообщения, предназначенные для всех диалоговых окон (dialog box), информационных окон (message box), полос прокрутки (scroll bar), или меню одного приложения (WH_MSGFILTER).
- Обработать или изменить все сообщения, предназначенные для всех диалоговых окон, информационных окон, полос прокрутки, или меню всей системы (WH_SYSMSGFILTER).
- Обработать или изменить все сообщения в системе (все виды сообщений), получаемые функциями **GetMessage** или **PeekMessage** (WH_GETMESSAGE).
- Обработать или изменить все сообщения (любого типа), посылаемые вызовом функции **SendMessage** (WH_CALLWNDPROC).

Использование хуков

- Записывать или проигрывать клавиатурные и мышиные события (WH_JOURNALRECORD, WH_JOURNALPLAYBACK).
- Обработать, изменить или удалить клавиатурные события (WH_KEYBOARD).
- Обработать, изменить или отменять события мыши (WH_MOUSE).
- Реагировать на определенные действия системы, делая возможным разработку приложений компьютерного обучения - computer-based training (WH_CBT).
- Предотвратить вызов другой функции-фильтра (WH_DEBUG).

Примеры использования хуков

- Добавить поддержку кнопки F1 для меню, диалоговых и информационных окон (WH_MSGFILTER).
- Обеспечить запись и воспроизведение событий мыши и клавиатуры, часто называемых макросами. Например, программа Windows Recorder использует хуки для записи и воспроизведения (WH_JOURNALRECORD, WH_JOURNALPLAYBACK).
- Следить за сообщениями, чтобы определить, какие сообщения предназначены определенному окну или какие действия генерирует сообщение (WH_GETMESSAGE, WH_CALLWNDPROC). Утилита Spy из Win32™ Software Development Kit (SDK) for Windows NT™ использует для этих целей хуки. Исходные тексты Spy можно найти в SDK.
- Симулировать мышиный и клавиатурный ввод (WH_JOURNALPLAYBACK). Хуки - единственный надежный способ симуляции этих действий. Если попытаться имитировать их через посылку сообщений, не будет происходить обновление состояния клавиатуры или мыши во внутренних структурах Windows, что может привести к непредсказуемому поведению. Если для воспроизведения клавиатурных или мышиных событий используются хуки, эти события обрабатываются в точности так, как и настоящий ввод с клавиатуры или мыши. Microsoft Excel использует хуки для реализации макрофункции SEND.KEYS.

Как пользоваться хуками

Чтобы пользоваться хуками, вам необходимо знать следующее:

- Как использовать функции Windows для добавления и удаления фильтрующих функций из очереди функций хука.
- Какие действия должна будет выполнить фильтрующая функция, которую вы устанавливаете.
- Какие существуют виды хуков, что они могут делать, и какую информацию (параметры) они передают вашей функции.

Функции Windows для работы с хуками

Приложения Windows используют функции

- **SetWindowsHookEx**
- **UnhookWindowsHookEx**
- **CallNextHookEx**

для управления очередью функций-фильтров хука.

SetWindowsHookEx

- Функция **SetWindowsHookEx** добавляет функцию-фильтр к хуку. Эта функция принимает четыре аргумента:
- Целочисленный код, описывающий хук, к которому будет прикреплена фильтрующая функция. Эти коды определены в WINUSER.H.
- Адрес функции-фильтра. Эта функция должна быть описана как экспортируемая включением ее в секцию **EXPORTS** файла определения приложения или библиотеки динамической линковки (DLL), или использованием соответствующих опций компилятора.
- Хэндл модуля, содержащего фильтрующую функцию. В Win32 , этот параметр должен быть NULL при установке хука на поток, но данное требование не является строго обязательным, как указано в документации. При установке хука для всей системы или для потока в другом процессе, нужно использовать хэндл DLL, содержащей функцию-фильтр.
- Идентификатор потока, для которого устанавливается хук. Если этот идентификатор ненулевой, установленная фильтрующая функция будет вызываться только в контексте указанного потока. Если идентификатор равен нулю, установленная функция имеет системную область видимости и может быть вызвана в контексте любого потока в системе. Приложение или библиотека могут использовать функцию GetCurrentThreadId для получения идентификатора текущего потока.

Область видимости хука

Хук	Область видимости
WH_CALLWNDPROC	Поток или вся система
WH_CBT	Поток или вся система
WH_DEBUG	Поток или вся система
WH_GETMESSAGE	Поток или вся система
WH_JOURNALRECORD	Только система
WH_JOURNALPLAYBACK	Только система
WH_FOREGROUNDIDLE	Поток или вся система
WH_SHELL	Поток или вся система
WH_KEYBOARD	Поток или вся система
WH_MOUSE	Поток или вся система
WH_MSGFILTER	Поток или вся система
WH_SYSMSGFILTER	Только система

Для любого данного типа хука, первыми вызываются хуки потоков, и только затем системные хуки.

Хуки потоков:

- Не создают лишней работы приложениям, которые не заинтересованы в вызове хука.
- Не помещают все события, относящиеся к хуку, в очередь (так, чтобы они поступали не одновременно, а одно за другим). Например, если приложение установит клавиатурный хук для всей системы, то все клавиатурные сообщения будут пропущены через фильтрующую функцию этого хука, оставляя неиспользованными системные возможности многопоточковой обработки ввода. Если эта функция прекратит обрабатывать клавиатурные события, система будет выглядеть зависшей, хотя на самом деле и не зависнет.
- Не требуют нахождения функции-фильтра в отдельной DLL. Все системные хуки и хуки для потоков в другом приложении должны находиться в DLL.

- **SetWindowsHookEx** возвращает хэндл установленного хука (тип HHOOK).
- Приложение или библиотека должны использовать этот хэндл для вызова функции **UnhookWindowsHookEx**.
- **SetWindowsHookEx** возвращает NULL если она не смогла добавить функцию к хуку.
- **SetWindowsHookEx** также устанавливает код последней ошибки в одно из следующих значений для индикации неудачного завершения функции.
- Windows сама заботится об организации очереди функций-фильтров не доверяя функциям хранение адресов следующих функций в очереди.

- **ERROR_INVALID_HOOK_FILTER:** Неверный код хука.
- **ERROR_INVALID_FILTER_PROC:** Неверная фильтрующая функция.
- **ERROR_HOOK_NEEDS_HMOD:** Глобальный хук устанавливается с параметром *hInstance*, равным NULL либо локальный хук устанавливается для потока, который не принадлежит данному приложению.
- **ERROR_GLOBAL_ONLY_HOOK:** Хук, который может быть только системным, устанавливается как потоковый.
- **ERROR_INVALID_PARAMETER:** Неверный идентификатор потока.
- **ERROR_JOURNAL_HOOK_SET:** Для регистрационного хука (journal hook) уже установлена фильтрующая функция. В любой момент времени может быть установлен только один записывающий или воспроизводящий хук. Этот код ошибки может также означать, что приложение пытается установить регистрационный хук в то время, как запущен хранитель экрана.
- **ERROR_MOD_NOT_FOUND:** Параметр *hInstance* в случае, когда хук является глобальным, не ссылался на библиотеку. (На самом деле, это значение означает лишь, что модуль User не смог обнаружить данный хэндл в списке модулей.)
- Любое другое значение: Система безопасности не позволяет установить данный хук, либо в системе закончилась память.

UnhookWindowsHookEx

- Для удаления функции-фильтра из очереди хука вызовите функцию **UnhookWindowsHookEx**.
- Эта функция принимает хэндл хука, полученный от **SetWindowsHookEx** и возвращает логическое значение, показывающее успех операции.
- На данный момент **UnhookWindowsHookEx** всегда возвращает TRUE.

Фильтрующие функции

- Фильтрующие (*хуковые*) функции - это функции, прикрепленные к хуку. Из-за того, что эти функции вызываются Windows, а не приложением, их часто называют *функциями обратного вызова* (callback functions).
- Все фильтрующие функции должны быть описаны следующим образом:
- LRESULT CALLBACK FilterFunc(int nCode, WPARAM wParam, LPARAM lParam)
- Все функции-фильтры должны возвращать **LONG**. Вместо *FilterFunc* должно стоять имя вашей фильтрующей функции.

Параметры

- Фильтрующие функции принимают три параметра: *nCode* (код хука), *wParam*, и *lParam*. Код хука - это целое значение, которое передает функции дополнительную информацию. К примеру, код хука может описывать событие, которое привело к срабатыванию хука.
- Вторым параметром функции-фильтра, *wParam*, имеет тип WPARAM, и третий параметр, *lParam*, имеет тип LPARAM. Эти параметры передают информацию фильтрующим функциям.
- У каждого хука значения *wParam* и *lParam* различаются. Например, фильтры хука WH_KEYBOARD получают в *wParam* виртуальный код клавиши, а в *lParam* - состояние клавиатуры на момент нажатия клавиши.
- Фильтрующие функции, прикрепленные к хуку WH_MSGFILTER получают в *wParam* значение NULL, а в *lParam* - указатель на структуру, описывающую сообщение.
- За полным описанием значений аргументов каждого типа хука обратитесь к Win32 SDK.

Вызов следующей функции в цепочке фильтрующих функций

- Когда хук уже установлен, Windows вызывает первую функцию в очереди, и на этом ее ответственность заканчивается. После этого функция ответственна за то, чтобы вызвать следующую функцию в цепочке.
- В Windows имеется функция **CallNextHookEx** для вызова следующего фильтра в очереди фильтров. **CallNextHookEx** принимает четыре параметра.
- Первый параметр - это значение, возвращенное функцией **SetWindowsHookEx**. В настоящее время Windows игнорирует это значение, но в будущем это может измениться.
- Следующие три параметра - *nCode*, *wParam*, и *lParam* - Windows передает дальше по цепочке функций.
- Windows хранит в своих внутренних структурах цепочку фильтрующих функций и следит за тем, какая функция вызывается в настоящий момент.
- При вызове **CallNextHookEx** Windows определяет следующую функцию в очереди и вызывает ее.

- Иногда функции-фильтры могут не пожелать передать обработку события другим фильтрам в той же цепочке. В частности, когда хук позволяет функции отменить событие и функция решает так поступить, она не должна вызывать **CallNextHookEx**. Когда фильтрующая функция модифицирует сообщение, она может решить не передавать его остальным функциям, ожидающим в очереди.
- Из-за того, что фильтры никак не сортируются при помещении их в очередь, вы не можете быть уверены, где находится ваша функция в любой момент времени кроме момента установки, когда ваша функция помещается в самое начало очереди. В результате, вы никогда не можете точно знать, что каждое событие в системе дойдет до вашего фильтра. Фильтрующая функция перед вашей функцией в цепочке - то есть функция, которая была установлена позже вашей - может не передать вам обработку события.

Хук WH_CALLWNDPROC

- Windows вызывает этот хук при каждом вызове функции **SendMessage**. Фильтрующей функции передается код хука, показывающий, была ли произведена посылка сообщения из текущего потока, а также указатель на структуру с информацией о сообщении.
- Структура CWPSTRUCT описана следующим образом:

```
typedef struct tagCWPSTRUCT {  
    LPARAM lParam;  
    WPARAM wParam;  
    DWORD message;  
    HWND hwnd; } CWPSTRUCT, *LPCWPSTRUCT;
```

- Фильтры могут обработать сообщение, но не могут изменять его. Сообщение затем отсылается той функции, которой и предназначалось. Этот хук использует значительное количество системных ресурсов, особенно, когда он установлен с системной областью видимости, поэтому используйте его только в целях отладки.

Хук WH_CBT

- Чтобы написать приложение для интерактивного обучения (CBT application), разработчик должен координировать его работу с работой приложения, для которого оно разрабатывается. Для достижения этой цели Windows предоставляет разработчикам хук WH_CBT. Windows передает фильтрующей функции код хука, показывающий, какое произошло событие, и соответствующие этому событию данные.
- Фильтр для хука WH_CBT должен знать о десяти хуковых кодах:
 - HCBT_ACTIVATE
 - HCBT_CREATEWND
 - HCBT_DESTROYWND
 - HCBT_MINMAX
 - HCBT_MOVESIZE
 - HCBT_SYSCOMMAND
 - HCBT_CLICKSKIPPED
 - HCBT_KEYSKIPPED
 - HCBT_SETFOCUS
 - HCBT_QS

Код HCBT_ACTIVATE

- Windows вызывает хук WH_CBT с этим кодом при активации какого-нибудь окна. Когда хук WH_CBT установлен как локальный, это окно должно принадлежать потоку, на который установлен хук. Если фильтр в ответ на это событие вернет TRUE, окно не будет активизировано.
- Параметр *wParam* содержит хэндл активизируемого окна. В *lParam* содержится указатель на структуру **CBTACTIVATESTRUCT**, которая описана следующим образом:
- `typedef struct tagCBTACTIVATESTRUCT {`
- `BOOL fMouse; // TRUE, если активация наступила в результате`
 `// мышинового клика; иначе FALSE.`
- `HWND hWndActive; // Содержит хэндл окна, активного`
 `// в настоящий момент.`
- `} CBTACTIVATESTRUCT, *LPCBTACTIVATESTRUCT;`

Код HCBT_CREATEWND

- Windows вызывает хук WH_CBT с этим при создании окна. Когда хук установлен как локальный, это окно должно создаваться потоком, на который установлен хук. Хук WH_CBT вызывается до того, как Windows пошлет новому окну сообщения WM_GETMINMAXINFO, WM_NCCREATE, или WM_CREATE. Таким образом, фильтрующая функция может запретить создание окна, вернув TRUE.
- В параметре *wParam* содержится хэндл создаваемого окна. В *lParam* - указатель на следующую структуру.
- ```
struct CBT_CREATEWND {
 struct tagCREATESTRUCT *lpcs; // Данные для создания нового окна.
 HWND hwndInsertAfter; // Хэндл окна, после которого будет
 // добавлено это окно (Z-order).
} CBT_CREATEWND, *LPCBT_CREATEWND;
```
- Функция-фильтр может изменить значение *hwndInsertAfter* или значения в *lpcs*.

# Код HCBT\_DESTROYWND

- Windows вызывает хук WH\_CBT с этим кодом перед уничтожением какого-либо окна. Если хук является локальным, это окно должно принадлежать потоку, на который установлен хук. Windows вызывает хук WH\_CBT до отправки сообщения WM\_DESTROY.
- Если функция-фильтр вернет TRUE, окно не будет уничтожено.
- Параметр *wParam* содержит хэндл уничтожаемого окна.
- В *lParam* находится 0L.

# Код HCBT\_MINMAX

- Windows вызывает хук WH\_CBT с этим кодом перед минимизацией или максимизацией окна. Когда хук установлен как локальный, это окно должно принадлежать потоку, на который установлен хук. Если фильтр вернет TRUE, действие будет отменено.
- В *wParam* передается хэндл окна, которое готовится к максимизации/минимизации. *lParam* содержит одну из SW\_\*-констант, определенных в WINUSER.H и описывающих операцию над окном.



# Код HCBT\_MOVE\_SIZE

- Windows вызывает хук WH\_CBT с этим кодом перед перемещением или изменением размеров окна, сразу после того, как пользователь закончил выбор новой позиции или размеров окна. Если хук установлен как локальный, это окно должно принадлежать потоку, на который установлен хук. Если фильтр вернет TRUE, действие будет отменено.
- В *wParam* передается хэндл перемещаемого/изменяемого окна.
- *lParam* содержит **LPRECT**, который указывает на новые координаты окна.

# Код HCBT\_SYSCOMMAND

- Windows вызывает хук `WH_CBT` с этим кодом во время обработки системной команды. Если хук установлен как локальный, окно, чье системное меню вызвало данное событие, должно принадлежать потоку, на который установлен хук. Хук `WH_CBT` вызывается из функции **DefWindowsProc**. Если приложение не передает сообщение `WH_SYSCOMMAND` функции **DefWindowsProc**, это хук не получит управление. Если функция-фильтр вернет `TRUE`, системная команда не будет выполнена.
- В *wParam* содержится системная команда (`SC_TASKLIST`, `SC_HOTKEY`, и так далее), готовая к выполнению. Если в *wParam* передается `SC_HOTKEY`, в младшем слове (`LOWORD`) *lParam* содержится хэндл окна, к которому относится горячая клавиша. Если в *wParam* передается любое другое значение и если команда системного меню была выбрана мышью, в младшем слове *lParam* будет находиться горизонтальная позиция, а в старшем слове (`HWORD`) - вертикальная позиция указателя мыши.
- Следующие системные команды приводят к срабатыванию этого хука изнутри **DefWindowProc**:

| SC_CLOSE      | Заккрыть окно.                                                       |
|---------------|----------------------------------------------------------------------|
| SC_HOTKEY     | Активировать окно, связанное с определенной горячей клавишей.        |
| SC_HSCROLL    | Горизонтальная прокрутка.                                            |
| SC_KEYMENU    | Выполнить команду меню по комбинации клавиш.                         |
| SC_MAXIMIZE   | Распахнуть окно.                                                     |
| SC_MINIMIZE   | Минимизировать окно.                                                 |
| SC_MOUSEMENU  | Выполнить команду меню по щелчку мыши.                               |
| SC_MOVE       | Переместить окно.                                                    |
| SC_NEXTWINDOW | Перейти к следующему окну.                                           |
| SC_PREVWINDOW | Перейти к предыдущему окну.                                          |
| SC_RESTORE    | Сохранить предыдущие координаты (контрольная точка - checkpoint).    |
| SC_SCREENSAVE | Запустить хранитель экрана.                                          |
| SC_SIZE       | Изменить размер окна.                                                |
| SC_TASKLIST   | Запустить или активировать Планировщик Задач (Windows Task Manager). |
| SC_VSCROLL    | Вертикальная прокрутка.                                              |

# Код HCBT\_CLICKSKIPPED

- Windows вызывает хук WH\_CBT с этим кодом при удалении события от мыши из входной очереди потока, в случае, если установлен хук мыши. Windows вызовет системный хук, когда из какой-либо входной очереди будет удалено событие от мыши и в системе установлен либо глобальный, либо локальный хук мыши. Данный код передается только в том случае, если к хуку WH\_MOUSE прикреплен фильтрующая функция. Несмотря на свое название, HCBT\_CLICKSKIPPED генерируется не только для пропущенных событий от мыши, но и в случае, когда событие от мыши удаляется из системной очереди. Его главное назначение - установить хук WH\_JOURNALPLAYBACK в ответ на событие мыши.
- В *wParam* передается идентификатор сообщения мыши - например, WM\_LBUTTONDOWN или любое из сообщений WM\_?BUTTON\*. *lParam* содержит указатель на структуру **MOUSEHOOKSTRUCT**, которая описана следующим образом:
- ```
typedef struct tagMOUSEHOOKSTRUCT {
```
- `POINT pt;` // Позиция курсора мыши в координатах экрана
- `HWND hwnd;` // Окно, получающее сообщение
- `UINT wHitTestCode;` // Результат проверки координат (hit-testing)
- `DWORD dwExtraInfo;` // Доп.информация о сообщении
- ```
} MOUSEHOOKSTRUCT, *LPMOUSEHOOKSTRUCT;
```

# Код HCBT\_KEYSKIPPED

- Windows вызывает хук WH\_CBT с этим кодом при удалении клавиатурного события из системной очереди, в случае, если установлен клавиатурный хук. Windows вызовет системный хук, когда из какой-либо входной очереди будет удалено событие от клавиатуры и в системе установлен либо глобальный, либо локальный клавиатурный хук.
- Данный код передается только в том случае, если к хуку WH\_KEYBOARD прикреплена фильтрующая функция. Несмотря на свое название, HCBT\_KEYSKIPPED генерируется не только для пропущенных клавиатурных событий, но и в случае, когда клавиатурное событие удаляется из системной очереди. Его главное назначение - установить хук WH\_JOURNALPLAYBACK в ответ на клавиатурное событие.
- В *wParam* передается виртуальный код клавиши - то же самое значение, что и в *wParam* функций **GetMessage** или **PeekMessage** для сообщений WM\_KEY\*. *lParam* содержит то же значение, что и *lParam* функций **GetMessage** или **PeekMessage** для сообщений WM\_KEY\*.

# WM\_QUEUESYNC

- Часто приложение интерактивного обучения (Computer Based Training application или *СВТ-приложение*) должно реагировать на события в процессе, для которого оно разработано. Обычно такими событиями являются события от клавиатуры или мыши. К примеру, пользователь нажимает на кнопку ОК в диалоговом окне, после чего СВТ-приложение желает послать главному приложению серию клавиатурных нажатий.
- СВТ-приложение может использовать хук мыши для определения момента нажатия кнопки ОК. После этого, СВТ-приложение должно выждать некоторое время, пока главное приложение не закончит обработку нажатия кнопки ОК (СВТ-приложение вряд ли хочет послать клавиатурные нажатия диалоговому окну).
- СВТ-приложение может использовать сообщение WM\_QUEUESYNC для определения момента окончания нужного действия. Слежение производится с помощью клавиатурного или мышинового хуков. Наблюдая за главным приложением с помощью хуков, СВТ-приложение узнает о наступлении необходимого события. После этого СВТ-приложение должно подождать окончания этого события, прежде чем приступить к выполнению ответных действий.

Для определения момента окончания обработки события, CBT-приложение делает следующее:

- 1. Ждет от Windows вызова хука WH\_CBT с кодом HCBT\_CLICKSKIPPED или HCBT\_KEYSKIPPED. Это происходит при удалении из системной очереди события, которое приводит к срабатыванию обработчика в главном приложении.
- 2. Устанавливает хук WH\_JOURNALPLAYBACK. CBT-приложение не может установить этот хук, пока не получит код HCBT\_CLICKSKIPPED или HCBT\_KEYSKIPPED.

Хук WH\_JOURNALPLAYBACK посылает CBT-приложению сообщение WM\_QUEUESYNC. Когда CBT-приложение получает такое сообщение, оно может выполнить необходимые действия, например, послать главному приложению серию клавиатурных нажатий.

# Код HCBT\_SETFOCUS

- Windows вызывает хук WH\_CBT с таким кодом, когда Windows собирается передать фокус ввода какому-либо окну. Когда хук установлен как локальный, это окно должно принадлежать потоку, на который установлен хук. Если фильтр вернет TRUE, фокус ввода не изменится.
- В *wParam* передается хэндл окна, получающего фокус ввода. *lParam* содержит хэндл окна, теряющего фокус ввода.



# Код HCBT\_QS

- Windows вызывает хук WH\_CBT с этим кодом когда из системной очереди удаляется сообщение WM\_QUEUESYNC, в то время как происходит изменение размеров или перемещение окна. Ни в каком другом случае этот хук не вызывается. Если хук установлен как локальный, это окно должно принадлежать потоку, на который установлен хук.
- Оба параметра - и *wParam*, и *lParam* - содержат ноль.

# Хук WH\_DEBUG

- Windows вызывает этот хук перед вызовом какой-либо фильтрующей функции. Фильтры не могут изменять значения, переданные этому хуку, но могут предотвратить вызов фильтрующей функции, возвратив ненулевое значение.
- В *wParam* передается идентификатор вызываемого хука, например, WH\_MOUSE. *lParam* содержит указатель на следующую структуру:

```
typedef struct tagDEBUGHOOKINFO
{
 DWORD idThread; // Идентификатор текущего потока
 LPARAM reserved;
 LPARAM lParam; // lParam для фильтрующей функции
 WPARAM wParam; // wParam для фильтрующей функции
 int code; } DEBUGHOOKINFO, * LPDEBUGHOOKINFO;
```

# WH\_FOREGROUNDIDLE

- Windows вызывает этот хук, когда к текущему потоку не поступает пользовательский ввод для обработки. Когда хук установлен как локальный, Windows вызывает его только при условии отсутствия пользовательского ввода у потока, к которому прикреплен хук. Данный хук является уведомительным, оба параметра - и *wParam*, и *lParam* - равны нулю.

# WH\_GETMESSAGE

- Windows вызывает этот хук перед выходом из функций **GetMessage** и **PeekMessage**.  
Фильтрующие функции получают указатель на структуру с сообщением, которое затем (вместе со всеми изменениями) посылается приложению, вызвавшему **GetMessage** или **PeekMessage**.
- В *lParam* находится указатель на структуру MSG:

# Регистрационные хуки

- Регистрационные хуки (journal hooks) используются для записи и воспроизведения событий. Они могут устанавливаться только как системные, и, следовательно, должны использоваться как можно реже. Эти хуки воздействуют на все приложения Windows; хотя десктоп и не позволяет такого другим хукам, регистрационные хуки могут записывать и воспроизводить последовательности событий и от десктопа, и для десктопа. Другой побочный эффект регистрационных хуков в том, что все системные входные очереди проходят через один поток, который установил такой хук.
- В Win32 предусмотрена специальная последовательность действий, с помощью которой пользователь может убрать регистрационный хук (например, в случае, если он зависил систему). Windows отключит записывающий или воспроизводящий регистрационный хук, когда пользователь нажмет CTRL+ESC, ALT+ESC, или CTRL+ALT+DEL.
- Windows оповестит приложение, установившее этот хук, посылкой ему сообщения WM\_CANCELJOURNAL.

# WM\_CANCELJOURNAL

- Это сообщение посылается с хэндлом окна, равным NULL, чтобы оно не попало в оконную процедуру. Лучший способ получить это сообщение - прикрепить к WH\_GETMESSAGE фильтрующую функцию, которая бы следила за входящими сообщениями. В документации по Win32 упоминается, что приложение может получить сообщение WM\_CANCELJOURNAL между вызовами функций **GetMessage** (или **PeekMessage**) и **DispatchMessage**.
- Хотя это и так, нет гарантий, что приложение будет вызывать эти функции, когда будет послано сообщение. Например, если приложение занято показом диалогового окна, главный цикл обработки сообщений не получит управление.
- Комбинации клавиш CTRL+ESC, ALT+ESC, и CTRL+ALT+DEL встроены в систему, чтобы пользователь всегда смог остановить регистрационный хук. Было бы неплохо, если каждое приложение, использующее регистрационные хуки, также предусматривало для пользователя способ остановки тотальной регистрации. Рекомендуемый способ - использовать код VK\_CANCEL (CTRL+BREAK).

# WH\_JOURNALRECORD

- Windows вызывает этот хук при удалении события из системной очереди. Таким образом, фильтры этого хука вызываются для всех мышиных и клавиатурных событий, кроме тех, которые проигрываются регистрационным хуком на воспроизведение. Фильтрующие функции могут обработать сообщение (то есть, записать или сохранить событие в памяти, на диске, или и там, и там), но не могут изменять или отменять его.
- Фильтры этого хука могут находиться и внутри DLL, и в .EXE-файле. В Win32 для этого хука реализован только код HC\_ACTION.

# HC\_ACTION

- Windows вызывает хук WH\_JOURNALRECORD с этим кодом при удалении события из системной очереди. Этот код сигнализирует фильтрующей функции о том, что это событие является нормальным. В lParam при этом передается указатель на структуру EVENTMSG. Обычная процедура записи состоит в сохранении всех пришедших хуку структур EVENTMSG в памяти или на диске.
- Структура EVENTMSG описана в WINDOWS.H следующим образом:

```
typedef struct tagEVENTMSG {
 UINT message;
 UINT paramL;
 UINT paramH;
 DWORD time;
 HWND hwnd;
} EVENTMSG;
```



- Элемент *message* является идентификатором сообщения, одним из значений WM\_\*. Значения *paramL* и *paramH* зависят от источника события - мышь это или клавиатура. Если это событие мыши, в *paramL* и *paramH* передаются координаты x и y события. Если это клавиатурное событие, в *paramL* находятся два значения: скан-код клавиши в HIBYTE и виртуальный код клавиши в LOBYTE, а *paramH* содержит число повторений. 15-й бит числа повторений служит индикатором дополнительной клавиши. В элементе *time* хранится системное время (наступления события), которое возвращается функцией **GetTickCount**. *hwnd* - это хэндл окна, получившего событие.
- Промежуток времени между событиями определяется сравнением элементов *time* этого события с элементом *time* последующего события. Разница во времени нужна для корректного проигрывания записанных событий.

# WH\_JOURNALPLAYBACK

- Этот хук используется для отправки Windows клавиатурных и мышиных сообщений таким образом, как будто они проходят через системную очередь. Основное назначение этого хука - проигрывание событий, записанных с помощью хука WH\_JOURNALRECORD, но его можно также с успехом использовать для отправки сообщений другим приложениям. Когда к этому хуку прикреплены фильтрующие функции, Windows вызывает первый фильтр в цепочке, чтобы получить событие. Windows игнорирует движения мыши, пока в системе установлен хук WH\_JOURNALPLAYBACK. Все остальные события от клавиатуры и мыши сохраняются до тех пор, пока у хука WH\_JOURNALPLAYBACK не останется функций-фильтров. Фильтры для этого хука могут располагаться как в DLL, так и в .EXE-файле. Фильтры этого хука должны знать о существовании следующих кодов:
- HC\_GETNEXT
- HC\_SKIP

# HC\_GETNEXT

- Windows вызывает WH\_JOURNALPLAYBACK с этим кодом, когда получает доступ к входной очереди потока. В большинстве случаев Windows посылает этот код несколько раз для одного и того же сообщения. В *lParam* фильтру передается указатель на структуру **EVENTMSG** (см. выше). Фильтрующая функция должна занести в эту структуру код сообщения *message*, *paramL*, и *paramH*. Обычно эти значения копируются из структур, записанных ранее с помощью хука WH\_JOURNALRECORD.
- Фильтрующая функция должна сообщить Windows когда нужно начинать обработку посланного сообщения. Windows необходимо для этого два значения: (1) период времени, на которое Windows должно задержать обработку сообщения; либо (2) точное время, когда это сообщение должно быть обработано. Обычно время ожидания обработки вычисляется как разница элементов *time* структуры **EVENTMSG** предыдущего сообщения и элемента *time* той же структуры текущего сообщения. Такой прием позволяет проигрывать сообщения на той же скорости, на которой они были записаны. Если сообщение необходимо проиграть немедленно, функция должна вернуть значение периода времен, равное нулю.

- Точное значение времени, в которое нужно обработать сообщение, обычно вычисляется сложением времени, которое Windows должна подождать до начала обработки сообщения и текущего системного времени, получаемого функцией **GetTickCount**. Для немедленного проигрывания сообщения используйте значение, возвращаемое функцией **GetTickCount**.
- Если система не находится в активном состоянии, Windows использует значения, переданные фильтром, для обработки события. Если система находится в активном состоянии, Windows проверяет системную очередь. Каждый раз, когда она это делает, Windows запрашивает то же самое событие с кодом HC\_GETNEXT. Каждый раз, когда функция-фильтр получает код HC\_GETNEXT, она должна вернуть новое значение времени ожидания, принимая во внимание время, прошедшее между вызовами функций. Элементы *message*, *paramH* и *paramL*, скорее всего, не потребуют изменений между вызовами.

# HC\_SKIP

- Windows вызывает хук WH\_JOURNALPLAYBACK после окончания обработки сообщения, полученного от WH\_JOURNALPLAYBACK. Это происходит в момент мнимого удаления события из системной очереди (мнимой, так как событие не находилось в системной очереди, а было сгенерировано хуком WH\_JOURNALPLAYBACK).
- Этот код хука сигнализирует фильтрующей функции о том, что событие, возвращенное фильтром во время вызова предыдущего HC\_GETNEXT, попало в приложение. Фильтрующая функция должна подготовиться вернуть следующее событие по приходу кода HC\_GETEVENT. Когда фильтрующая функция определяет, что больше нечего проигрывать, она должна удалиться из цепочки фильтров хука во время обработки кода HC\_SKIP.

# WH\_KEYBOARD

- Windows вызывает этот хук когда функции **GetMessage** или **PeekMessage** собираются вернуть сообщения WM\_KEYUP, WM\_KEYDOWN, WM\_SYSKEYUP, WM\_SYSKEYDOWN, или WM\_CHAR. Когда хук установлен как локальный, эти сообщения должны поступать из входной очереди потока, к которому прикреплен хук. Фильтрующая функция получает виртуальный код клавиши и состояние клавиатуры на момент вызова клавиатурного хука. Фильтры имеют возможность отменить сообщение. Фильтрующая функция, прикрепленная к этому хуку, должна знать о существовании следующих кодов:
  - HS\_ACTION
  - HS\_NOREMOVE

- **HC\_ACTION**
- Windows вызывает хук WH\_KEYBOARD с этим кодом при удалении события из системной очереди.
- **HC\_NOREMOVE**
- Windows вызывает хук WH\_KEYBOARD с этим кодом, когда клавиатурное сообщение не удаляется из очереди, потому что приложение вызвало функцию **PeekMessage** с параметром PM\_NOREMOVE. При вызове хука с этим кодом не гарантируется передача действительного состояние клавиатуры. Приложение должно знать о возможности возникновения подобной ситуации.

# WH\_MOUSE

- Windows вызывает этот хук после вызова функций **GetMessage** или **PeekMessage** при условии наличия сообщения от мыши. Подобно хуку WH\_KEYBOARD фильтрующие функции получают код - индикатор удаления сообщения из очереди (HC\_NOREMOVE), идентификатор сообщения мыши и координаты x и y курсора мыши.
- Фильтры имеют возможность отменить сообщение. Фильтры для этого хука должны находиться в DLL.



# WH\_MSGFILTER

- Windows вызывает этот хук, когда диалоговое окно, информационное окно, полоса прокрутки или меню получают сообщение, либо когда пользователь нажимает комбинацию клавиш ALT+TAB (или ALT+ESC) при активном приложении, установившем этот хук. Данный хук устанавливается для конкретного потока, поэтому его безопасно размещать как в самом приложении, так и в DLL. Фильтрующая функция этого хука получает следующие коды:
- MSGF\_DIALOGBOX: Сообщение предназначено либо диалоговому, либо информационному окну.
- MSGF\_MENU: Сообщение предназначено меню.
- MSGF\_SCROLLBAR: Сообщение предназначено полосе прокрутки.
- MSGF\_NEXTWINDOW: Происходит переключение фокуса на следующее окно.
- В WINUSER.H определено больше MSGF\_-кодов, но в настоящее время они не используются хуком WH\_MSGFILTER.
- В *IParam* передается указатель на структуру, содержащую информацию о сообщении. Хуки WH\_SYSMMSGFILTER вызываются перед хуками WH\_MSGFILTER. Если какая-нибудь из фильтрующих функций хука WH\_SYSMMSGFILTER возвратит TRUE, хуки WH\_MSGFILTER не будут вызваны.

# WH\_SHELL

- Windows вызывает этот хук при определенных действиях с окнами верхнего уровня - top-level windows (то есть, с окнами, не имеющими владельца). Когда хук установлен как локальный, он вызывается только для окон, принадлежащих потоку, установившему хук. Этот хук является информирующим, поэтому фильтры не могут изменять или отменять событие. В *wParam* передается хэндл окна; параметр *lParam* не используется. Для данного хука в WINUSER.H определены три кода:
- HSHELL\_WINDOWCREATED: Windows вызывает хук WH\_SHELL с этим кодом при создании окна верхнего уровня. Когда фильтр получает управление, это окно уже создано.
- HSHELL\_WINDOWDESTROYED: Windows вызывает хук WH\_SHELL с этим кодом перед удалением окна верхнего уровня.
- HSHELL\_ACTIVATESHELLWINDOW: На данный момент этот код не используется.

# WH\_SYSMSGFILTER

- Этот хук идентичен хуку WH\_MSGFILTER за тем исключением, что он имеет системную область видимости. Windows вызывает этот хук, когда диалоговое окно, информационное окно, полоса прокрутки или меню получает сообщение, либо когда пользователь нажимает комбинации клавиш ALT+TAB или ALT+ESC. Фильтр получает те же коды, что и фильтры хука WH\_MSGFILTER.
- В *lParam* передается указатель на структуру, содержащую информацию о сообщении. Хуки WH\_SYSMSGFILTER вызываются до хуков WH\_MSGFILTER. Если любая из фильтрующих функций хука WH\_SYSMSGFILTER вернет TRUE, фильтры хука WH\_MSGFILTER не будут вызваны.