**Федеральное государственное автономное образовательное учреждение**

**Высшего профессионального образования**

Санкт-Петербургский политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №6**

«Обработка событий в Windows»

Работу выполнил студент гр. № 53501/3 Цыганов А.А.

Работу принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Душутина Е.В.

Санкт-Петербург

2016

# Цель работы:

В данной лабораторной работе необходимо изучить механизмы и способы обработки событий в среде Windows.

Задачи:

* Написать динамическую библиотеку, предоставляющие функции языка С++ для создания ловушек в ОС Windows и их обработки (функции фильтра) для получения сигналов с клавиатуры и компьютерной мыши.
* Написать приложение, которое будет использовать динамическую библиотеку.
* Провести эксперименты по обработке событий нажатия мыши и клавиш клавиатуры.

# Ход работы:

Для работы была использована среда разработки Visual Studio 2015 Enterprise в ОС Windows 10 Pro.

Ловушка (hook) - это механизм, который позволяет производить мониторинг сообщений системы и обрабатывать их до того, как они достигнут целевой оконной процедуры. Приложение может использовать хуки как для получение информации о событиях в системе, так и для перехвата событий и измерения хода работы системы. Хуки, особенно глобальные, могут замедлять работу приложений.

В операционной системе Window 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. 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.

Для установки и доступа к фильтрующим функциям приложения используют функции **SetWindowsHookEx** и **UnhookWindowsHookEx**.

Функция **SetWindowsHookEx** добавляет функцию-фильтр к хуку.

HHOOK WINAPI SetWindowsHookEx(

\_In\_ int       idHook, *// тип ловушки*

\_In\_ HOOKPROC  lpfn, *// указатель на подключаемую функцию callback*

\_In\_ HINSTANCE hMod, // *дескриптор динамической библиотеки, содержащей функцию, переданную в предыдущем параметре*

\_In\_ DWORD     dwThreadId // *идентификатор потока, с которым должна быть связана ловушка (если передается NULL, то ловушка связана со всеми существующими потоками).*

);

Для удаления функции-фильтра из очереди хука вызовите функцию **UnhookWindowsHookEx**. Эта функция принимает обработчик хука, полученный от **SetWindowsHookEx** и возвращает логическое значение, показывающее результат операции.

Фильтрующие 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 описываются следующим образом:

|  |
| --- |
| LRESULT CALLBACK FilterFunc(int nCode, WPARAM wParam, LPARAM lParam); |

Любая функция(callback)-ловушка должна иметь следующий набор аргументов:

* Int nCode – код хука. Используется для определения функцией хука действия, которое необходимо исполнить.
* WPARAM wParam – параметр, определяемый кодом хука.
* LPARAM lParam – параметр, определяемый кодом хука.

В ходе работы будет осуществлён перехват событий клавиатуры и мышки, у которых типы хуков:

* WH\_KEYBOARD, WH\_KEYBOARD\_LL - события клавиатуры.
* WH\_MOUSE, WH\_MOUSE\_LL - события мыши.

# Создание динамической библиотеки

Разработка динамической библиотеки средствами Microsoft Visual Studio 2015. В библиотеке реализовать класс AlexWinHook, позволяющий установить и снять хуки, а также содержащий callback-функции для обработки событий клавиатуры и мыши.

Ниже приведен исходный код библиотеки класса AlexWinHook:

AlexWinHook.h

// Приведенный ниже блок ifdef - это стандартный метод создания макросов, упрощающий процедуру

// экспорта из библиотек DLL. Все файлы данной DLL скомпилированы с использованием символа ALEXWINHOOKLIB\_EXPORTS,

// в командной строке. Этот символ не должен быть определен в каком-либо проекте

// использующем данную DLL. Благодаря этому любой другой проект, чьи исходные файлы включают данный файл, видит

// функции ALEXWINHOOKLIB\_API как импортированные из DLL, тогда как данная DLL видит символы,

// определяемые данным макросом, как экспортированные.

#ifdef ALEXWINHOOKLIB\_EXPORTS

#define ALEXWINHOOKLIB\_API \_\_declspec(dllexport)

#else

#define ALEXWINHOOKLIB\_API \_\_declspec(dllimport)

#endif

#include<windows.h>

typedef void(\*onAlexWinHook\_keyboard)(DWORD keyCode);

typedef void(\*onAlexWinHook\_mouse)(WPARAM wParam);

// Этот класс экспортирован из AlexWinHookLib.dll

class ALEXWINHOOKLIB\_API CAlexWinHookLib {

private:

// хук для клавиатуры

static LRESULT CALLBACK CBKeyboard(int nCode, WPARAM wParam, LPARAM lParam);

//хук для мышки

static LRESULT CALLBACK CBMouse(int nCode, WPARAM wParam, LPARAM lParam);

public:

// устанавливает хук на отслеживание нажатия клавиш

void keyboardHook(onAlexWinHook\_keyboard hook);

// устанавливает хук на отслеживание нажатия мыши

void mouseHook(onAlexWinHook\_mouse hook);

};

У класса два открытых метода для создания ловушек на клавиатуру keyboardHook() и на нажатие мышки mouseHook(). Параметры эти функций принимают указатель на функцию, через которую можно определить факт перехвата. Класса CAlexWinhookLib экспортирован из библиотеки. Ниже приведена реализация методов класса с функциями WinAPI для установки ловушки.

AlexWinHook.m

#include "stdafx.h"

#include <iostream>

#include "AlexWinHookLib.h"

#include <Windows.h>

using namespace std;

static HHOOK mMouseHook, mKeyboardHook;

static onAlexWinHook\_keyboard mcbHook\_keyboard;

static onAlexWinHook\_mouse mcbHook\_mouse;

// хук для клавиатуры

LRESULT CALLBACK CAlexWinHookLib::CBKeyboard(int nCode, WPARAM wParam, LPARAM lParam)

{

// проверка на то, что произошло именно нажатие клавиш

if (wParam == WM\_KEYDOWN || wParam == WM\_SYSKEYDOWN)

{

//получаем структуру, описывающую нажатую клавишу

PKBDLLHOOKSTRUCT pKey = (PKBDLLHOOKSTRUCT)lParam;

//извлекаем и выводим коде клавиши

// отдельная обработка для VK\_PRIOR (page up)

DWORD d = pKey->vkCode;

mcbHook\_keyboard(d);

}

return CallNextHookEx(mKeyboardHook, nCode, wParam, lParam);

}

//хук для мышки

LRESULT CALLBACK CAlexWinHookLib::CBMouse(int nCode, WPARAM wParam, LPARAM lParam)

{

MSLLHOOKSTRUCT \* pMouseStruct = (MSLLHOOKSTRUCT \*)lParam;

if (nCode == 0 && pMouseStruct ) {

mcbHook\_mouse(wParam);

}

return CallNextHookEx(mKeyboardHook, nCode, wParam, lParam);

}

void CAlexWinHookLib::keyboardHook(onAlexWinHook\_keyboard hook)

{

mcbHook\_keyboard = hook;

// получаем дескриптор модуля, запустившего процесс

HMODULE hInstance = GetModuleHandle(NULL);

mKeyboardHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, CBKeyboard, hInstance, NULL);

//ожидает получение сообщения из очереди сообщений потока.

MSG msg;

GetMessage(&msg, 0, 0, 0);

//отвязывает mKeyboardHook

UnhookWindowsHookEx(mKeyboardHook);

}

// устанавливает хук на отслеживание нажатия мыши

void CAlexWinHookLib::mouseHook(onAlexWinHook\_mouse hook)

{

mcbHook\_mouse = hook;

// получаем дескриптор модуля, запустившего процесс

HMODULE hInstance = GetModuleHandle(NULL);

mMouseHook = SetWindowsHookEx(WH\_MOUSE\_LL, CBMouse, hInstance, NULL);

//ожидает получение сообщения из очереди сообщений потока.

MSG msg;

GetMessage(&msg, NULL, NULL, NULL);

//отвязывает mMouseHook

UnhookWindowsHookEx(mMouseHook);

}

Обработка установленных ловушек на клавиатуру и мышку, происходит в функция callbacl – соответственно LRESULT CALLBACK CAlexWinHookLib::CBKeyboard(int nCode, WPARAM wParam, LPARAM lParam) и LRESULT CALLBACK CAlexWinHookLib::CBMouse(int nCode, WPARAM wParam, LPARAM lParam).

При перехвате события мышки, получаем ссылку на структуру MSLLHOOKSTRUCT, из которой можно получить подробную информация о событии. Структура MSLLHOOKSTRUCT приведена ниже:

typedef struct tagMSLLHOOKSTRUCT {

POINT     pt;

DWORD     mouseData;

DWORD     flags;

DWORD     time;

ULONG\_PTR dwExtraInfo;

} MSLLHOOKSTRUCT, \*PMSLLHOOKSTRUCT, \*LPMSLLHOOKSTRUCT;

Аналогично для получения информации о событии клавиатуры – структура KBDLLHOOKSTRUCT. Ниже приведена стурктура KBDLLHOOKSTRUCT:

typedef struct tagKBDLLHOOKSTRUCT {

DWORD     vkCode;

DWORD     scanCode;

DWORD     flags;

DWORD     time;

ULONG\_PTR dwExtraInfo;

} KBDLLHOOKSTRUCT, \*PKBDLLHOOKSTRUCT, \*LPKBDLLHOOKSTRUCT;

Для тестирования библиотеки напишем консольное приложение, в котором реализуем экземпляр класса CAlexWinHookLib и соответствующие функции callback. Ниже приведен исходный код приложения:

// AlexWinHookTest.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include "AlexWinHookLib.h"

#include <iostream>

using namespace std;

void alexWinHook\_mouse(WPARAM wParam)

{

LPCTSTR key;

switch (wParam) {

case WM\_LBUTTONUP:

key = L"WM\_LBUTTONUP";

break;

case WM\_LBUTTONDOWN:

key = L"WM\_LBUTTONDOWN";

break;

case WM\_RBUTTONUP:

key = L"WM\_RBUTTONUP";

break;

case WM\_RBUTTONDOWN:

key = L"WM\_RBUTTONDOWN";

break;

}

MessageBox(NULL, key, L"Mouse clicked", NULL);

}

void alexWinHook\_keyboard(DWORD keyCode)

{

LPCTSTR key;

if (keyCode == VK\_F1) // для примера определение нажатия на кнопку F1

key = L"F1";

else

key = L"Some botton click";

MessageBox(NULL, key, L"Keybord pressed", NULL);

}

int main()

{

cout << "Windows hook Keyboard, mouse." << endl;

CAlexWinHookLib l = CAlexWinHookLib();

l.keyboardHook(alexWinHook\_keyboard);

l.mouseHook(alexWinHook\_mouse);

return 0;

}

Все события с клавиатурой и мышкой будут пользователю отображены в всплывающем модальном окне. При запуске программы происходит установка ловушки и на клавиатуру, и на мышку. Реализованы функции callback alexWinHook\_mouse() и alexWinHook\_keyboard() функции, которые вызываются из динамической библиотеки при срабатывании нажатия на кнопку.

Сборка решения средствами Visual Studio 2015 прошла успешно:

1>------ Сборка начата: проект: AlexWinHookLib, Конфигурация: Debug Win32 ------

1> stdafx.cpp

1> AlexWinHookLib.cpp

1> Создание кода...

1> dllmain.cpp

1> Создается библиотека C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\Lab7\Debug\AlexWinHookLib.lib и объект C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\Lab7\Debug\AlexWinHookLib.exp

1> AlexWinHookLib.vcxproj -> C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\Lab7\Debug\AlexWinHookLib.dll

2>------ Сборка начата: проект: AlexWinHookTest, Конфигурация: Debug Win32 ------

2> stdafx.cpp

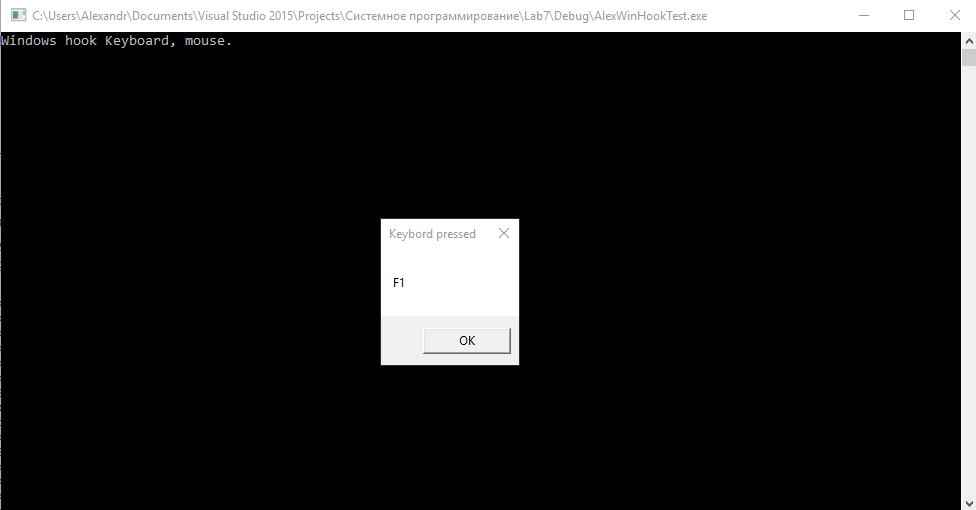
2> AlexWinHookTest.cpp

2> Создание кода...

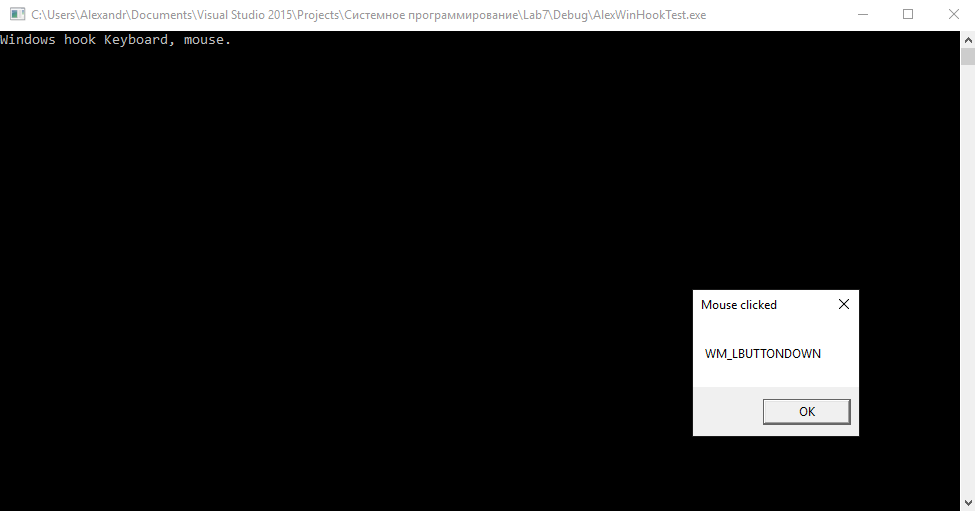
2> AlexWinHookTest.vcxproj -> C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\Lab7\Debug\AlexWinHookTest.exe

========== Сборка: успешно: 2, с ошибками: 0, без изменений: 0, пропущено: 0 ==========

Далее представлены результаты запуска программы. При нажатии на клавиатуру или мышку происходит перехват и показывается пользователю сообщение. Ниже приведён пример работы программы и нажатия на кнопку F1.



Далее приведен пример с перехватом нажатия кнопки мышки:



Используемые источники

1. https://msdn.microsoft.com/ru-ru/library/windows/desktop/ms644970(v=vs.85).aspx