VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE K MONITOROVÁNÍ UDÁLOSTÍ OS WINDOWS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MAREK BUKOVSKÝ
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE K MONITOROVÁNÍ UDÁLOSTÍ OS WINDOWS

APPLICATION FOR MONITORING EVENTS OF OS WINDOWS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MAREK BUKOVSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ GRULICH
SUPERVISOR

BRNO 2008

Abstrakt

Operačný systém predstavuje základ pre prácu s počítačom. Činnosti, ktoré sa odohrávajú v pozadí, vo vnútri operačného systému sú väčšinou pre bežného užívateľa veľkou neznámou. Práca popisuje práve aplikáciu pre odhalenie činností operačného systému Windows. Slúži k monitoravaniu udalostí systému. Popisuje aké techniky boli použité pri implementácii a samotnú implementáciu jednotlivých častí. Práca taktiež popisuje vytvorenú aplikáciu, jej jednotlivé časti. Vysvetľuje teóriu monitorovacích techník a všetkých prvkov operačného systému, ktoré boli k tomu použité.

Klíčová slova

Windows háky, dynamicky pripájaná knižnica, procesy, medziprocesová komunikácia, Windows správy, prostredie .net

Abstract

Operation system represents basis for work with computer. Activities, which are in the background, inside operation system are mostly for common user big unknown. This work describes exactly application used to reveal background activities of operation system Windows. Application serves for monitoring system events. This work discribes also techniques which were used for implementation and implementation of individual parts itself. Work also describes created application, its individual parts. Explains theory of monitoring techniques and all components of operation system, which were used for monitoring itself.

Keywords

Windows hooks, dynamic-link library, processes, inter-process communication, Windows messages, .net platform

Citace

Marek Bukovský: Aplikace k monitorování událostí OS Windows, bakalářská práce, Brno, FIT VUT v Brně, 2008

Aplikace k monitorování událostí OS Windows

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Lukáša Grulicha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Marek Bukovský
12.05.2008

Obsah

Kapitola 1

Úvod

Dnešný svet je plný informačných technológií a počítačov. Bežný užívateľ pracuje v operačnom systéme bez toho, aby vedel čo sa odohráva na pozadí. Pre tých väčších znalcov existujú programy, ktoré umožňujú monitorovať systémové udalosti a umožniť užívateľovi presne sledovať činnosť operačného systému.

K takýmto programom patrí aj táto práca. Konkrétne sa jedná o monitorovanie udalostí operačného systému Windows. Práca objasňuje činnosť operačného systému a príslušných častí týkajúcich sa monitorovania udalostí OS Windows. Sú vysvetlené princípy komunikácie aplikácií v rámci operačného systému a komunikácia operačného systému s aplikáciami. V práci je uvedený spôsob použitý pre monitorovanie operačného systému.

Úlohou bolo vytvoriť aplikáciu, ktorá by vytvárala záznam o vzniknutých udalostiach. Umožňovala by ukladanie záznamu do externých súborov na disk. Ďalej by bolo možné nastaviť filter správ, ktoré majú byť monitorované. Taktiež pre jednoduchú obsluhu aplikácie vytvoriť jednoduché užívateľské rozhranie. Vďaka záznamu udalostí operačného systému užívateľ vie, čo sa odohráva na pozadí, v akom slede sa odohrali jednotlivé udalosti, ako na ne systém zareagoval, prípadne ako boli následne spracované koncovou aplikáciou.

Jednotlivé kapitoly práce podrobne popisujú ako bola daná práca vytvorená, aké prvky boli využité, vzniknuté problémy a ich riešenie.

Kapitola 2

Monitorovanie systémových udalostí a Windows hooks

Kapitola popisuje čo sú to systémové udalosti, reakciu operačného systému na vzniknuté udalosti, ich následné spracovanie, medziprocesovú komunikáciu, využitie metódy takzvaného. *hákovania* (hooking) pre odchytávanie systémových udalostí.

2.1 Udalosti OS Windows

V tejto sekcii bude popísaná činnosť operačného systému. V tomto prípade sa bude jednať o operačný systém Windows od spoločnosti Microsoft. Konkrétne bude vysvetlené čo sú to udalosti, ich vznik a následné spracovanie systémom. Taktiež bude vysvetlené čo sú to správy a ako ich operačný systém využíva ku vnútornej komunikácii.

2.1.1 Udalosť

Udalosť vzniká v operačnom systéme na určitý podnet. Systém na vzniknutú udalosť zareaguje a spracuje ju. Podnet pre udalosť môže byť externý alebo interný. Externé podnety sú väčšinou od užívateľov. To znamená napríklad stlačenie klávesy na klávesnici, pohyb myšou, záznam zvuku z mikrofónu, dotyk na doske dotykového displeja a iné formy komunikácie užívateľa s počítačom viz. [8]. Interné udalosti sú udalosti generované samotným systémom ako napr. prekreslenie užívateľskej plochy v prípade zmeny zobrazovaných prvkov.

2.1.2 Správy OS Windows

Operačný systém Windows je založený na vnútornej komunikácii pomocou správ. Správa je vygenerovaná ako reakcia na udalosť. Správa nesie informáciu o tom aká udalosť vznikla, kto ju má spracovať a ďalšie špecifické parametre, ktoré závisia od odoslanej správy. Napríklad pri stlačení klávesy "A" vzniká udalosť, na ktorú systém zareaguje tak, že odošle správu. Správa je zaslaná aplikácii, ktorá je aktívna a zachytáva udalosti klávesnice, čiže má tzv. *Keyboard focus*. Správa obsahuje ako špecifické parametre hodnotu znaku "A". Každá správa v systéme má svoj jedinečný identifikátor, na základe ktorého systém vie o akú správu sa jednalo.

Väčšina operačných systémov obsahuje *rad správ* (message queue). Je to prostriedok, ktorý umožňuje operačnému systému komunikáciu medzi rôznymi vláknami a procesmi. Vysvetlenie čo sú procesy a vlákna a ich podrobnejší popis bude v sekcii 2.4. Rad správ

poskytuje asynchrónny komunikačný protokol. V praxi to znamená, že vysielač a prijímač správ nemusia vzájomne komunikovať v rovnakom čase. V operačnom systéme Windows existuje rad správ využívaný aplikáciami. Bežiaca aplikácia následne sleduje rad správ a vyberá si z neho správy a postupne ich spracuje.

Každá aplikácia bežiaca pod operačným systémom Windows musí obsahovať tzv. *slučku správ* (message loop). Je to nekonečný cyklus, kde bežiaca aplikácia neustále kontroluje či jej bola zaslaná správa. Využíva pri tom už spomenutý rad správ. Slučka správ v OS Windows funguje celkom jednoducho. Aplikácie OS Windows obsahujú dve funkcie, ktoré sú určené pre spracovanie správ. Jedna funkcia je predvolená pre spracovanie všetkých správ. Jedná sa o štandardné spracovanie správ. Ak chce užívateľ nejakú správu spracovať, tak nadefinuje jej spracovanie v druhej funkcii. Druhá funkcia je užívateľom definovaná tzv. *callback* funkcia pre spracovanie správ. Callback funkcie sú funkcie, ktoré sú ako odozva na určitý podnet, ako odpoveď, ako spätné zavolanie funkcie. Na jej konci je zvyčajne zavolaná predvolená funkcia, aby boli spracované všetky správy, aj tie, ktoré užívateľ nepotrebuje. Slučka správ v operačnom systéme Windows má nasledujúci sled operácií. V prvom kroku vyberie správu z radu. Pokiaľ by sa jednalo o správu, ktorá by znamenala ukončenie aplikácie, tak sa touto správou ukončí slučka správ. To sa deje automaticky vďaka funkcii, pomocou ktorej vyberáme správy z radu správ. V druhom kroku nasleduje preklad správy. A v nasledujúcom poslednom kroku sa zavolá funkcia, ktorá odošle správu už spomenutej funkcii pre spracovanie správ. Aby sa správa doručila funkcii pre spracovanie správ v správnej aplikácii a aby aplikácia nespracovala cudzie správy, tak funkcia v poslednom kroku odošle správu aplikácii podľa identifikácie okna, ktorá je súčasťou správy. Znalosti ohľadom správ, slučky správ a radu správ boli získané zo stránok wikipedie [10] a [11].

2.2 Windows hooks

Jedná sa o mechanizmus spracovania správ, kde aplikácia môže nainštalovať podprogram na monitorovanie prenosu správ v systéme predtým, ako sú doručené cieľovej aplikácii. K nainštalovaniu, zavedeniu tohto mechanizmu sa využívajú systémové funkcie. Windows hooks (Windows háky) majú tendenciu spomaliť činnosť operačného systému a znížiť jeho výkon. Hákmi sa nazývajú, pretože sa tak povediac zaháknu medzi operačný systém a aplikáciu. Háky môžu byť rôzneho charakteru a to globálne alebo lokálne. Ďalšie podrobnosti ohľadom hákov budú rozobraté v nasledujúcich sekciách.

2.2.1 Hákovanie (Hooking)

Hákovanie je programovacia technika, ktorá využíva tzv. *háky* (hooks). Háky vytvoria reťaz procedúr, ktoré spracujú vzniknuté udalosti. Čiže sa použijú ako tzv. *event handler*. Zaviesť háky do prevádzky znamená, použiť hákom definovanú procedúru pre spracovanie vzniknutej udalosti. Činnosť tejto techniky je nasledujúca. Po vzniku udalosti, ktorá má byť spracovaná nejakou aplikáciou sa najskôr predá riadenie reťazcu procedúr, ktoré sú zavedené pomocou hákov. Nový hák si zaregistruje svoju vlastnú adresu procedúry, ktorá ju spracuje. V určitom bode sa očakáva, že daná procedúra predá riadenie pôvodnej aplikácii. Najčastejšie to býva na konci. Ak by nenastalo predanie riadenia ďalšiemu háku, alebo pôvodnej aplikácii mohlo by dôjsť k porušeniu reťazca spracovania vzniknutej udalosti. Zrušiť háky znamená, použiť pôvodné procedúry pre spracovanie udalostí.

Háky môžu byť použité rôznym spôsobom. Napríklad je možné ich použiť pre *ladenie* (debugging) aplikácií. A to vďaka tomu, že pri zavedení správnych hákov je možné

presne povedať, akým spôsobom aplikácia komunikuje s operačným systémom a aké správy mu posiela. Hlavnou výhodou tohto ladenia je, že je možné ladiť už preložené aplikácie. Ďalšie využitie pre háky je rozšírenie funkcionality už existujúcich aplikácií. Háky môžu byť aj zneužité pre nekalé účely. A to hlavne pri aplikáciách, ktoré majú väčšinou škodlivé správanie. Jedná sa o aplikácie, o ktorých užívateľ nemá vedieť a háky mu umožnia skryť ich existenciu odchytávaním správ, ktoré by mohli užívateľa upozorniť na ich existenciu. Jedno z možných využití je napríklad aj monitorovanie prevádzky správ v operačnom systéme. Je to podstata tejto bakalárskej práce. Monitorovanie znamená len odchytávať správy a uchovávať ich názov, prípadne ďalšie parametre týkajúce sa správy. Žiadne úpravy prechádzajúcich správ sa nedejú a vždy sa preposielajú ďalej, aby sa zachoval reťazec spracovania udalosti.

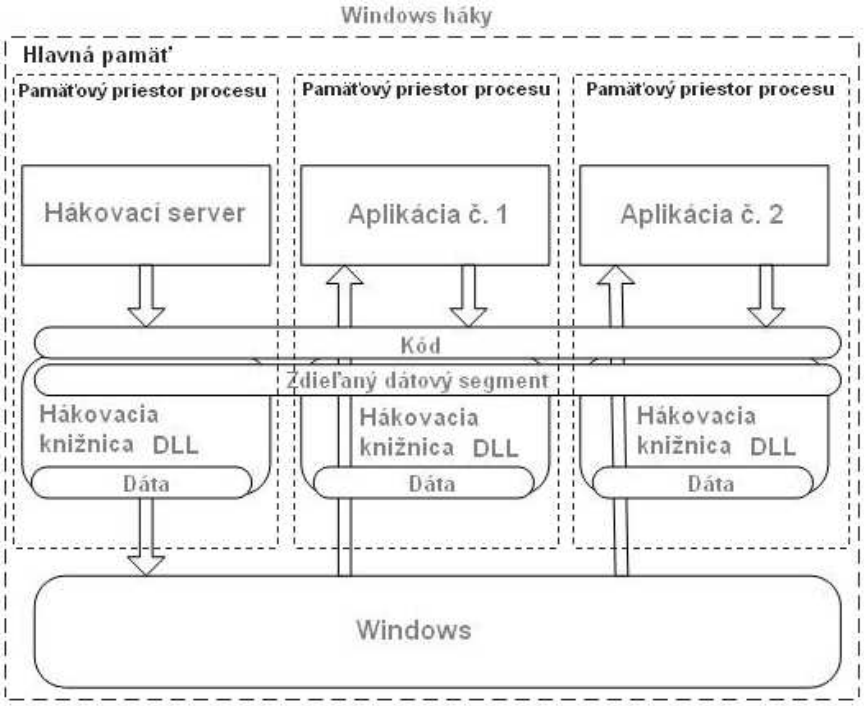
Ako už bolo spomenuté háky môžu mať lokálny alebo globálny charakter. Lokálny charakter znamená, že definovaný hák odchytáva iba správy určené vláknu, na ktorom beží daná aplikácia. Globálny charakter znamená, že aplikácia zavedie hák pre celý systém. Čiže odchytáva správy určené pre všetky aplikácie a aj správy, ktoré sú určené iba pre samotný systém.

Techniky pre odchytávanie a monitorovanie činnosti systému

1. Existuje spôsob, ktorý je veľmi podobný hákovaniu. Je to za pomoci systémových registrov. Vo Windows registroch existuje položka, ktorá nastavuje aké knižnice sa majú zaviesť pri spúšťaní aplikácií. Pokiaľ je do tohto tzv. *kľúča* (registry key) nastavená užívateľom definovaná knižnica, tak sa zavedie pri spúšťaní každej aplikácie. Je to však neškodný spôsob. A navyše môže byť použitý iba pre operačné systémy Windows NT, Windows 2000 a systémy vychádzajúce z technologie NT. Taktiež obmedzenie je, že je to použiteľné iba pre aplikácie, ktoré využívajú systémovú knižnicu *user32.dll*.

2. Samotné hákovanie je veľmi obľúbený spôsob monitoronia prevádzky správ, prípadne ich úprav. Ako už bolo spomenuté, háky sa zavedú za pomoci systémových funkcií. Funkcia pre zavedenie obsahuje potrebné informácie pre zavedenie konkrétneho háku. Existuje viac druhov hákov a užívateľ si môže vybrať, ktorý hák sa hodí pre jeho potreby. Funkcia pre zavedenie háku obsahuje typ háku, ukazovateľ na funkciu, ktorá bude zavolaná vždy keď vznikne udalosť, ktorú hák sleduje a modul v ktorom sa táto funkcia nachádza. Ďalej obsahuje informácie o tom či sa jedná o globálny alebo lokálny hák. Táto informácia je v podobe identifikátora vlákna, ktoré sa má monitorovať. Ak by sa jednalo o lokálny hák, tak sa použije ID vlákna, ktoré sa bude monitorovať. Pokiaľ by šlo o globálny hák použije sa 0, čo znamená, že sa budú monitorovať všetky vlákna na tej istej pracovnej ploche ako vlákno, v ktorom beží samotné monitorovanie. Práve z tohto dôvodu musí byť funkcia, ktorá bude spracovávať vzniknutú udalosť umiestnená do samotnej knižnice. Jedná sa o tzv. *dynamicky pripájanú knižnicu* (dynamic-link library) DLL. Dôvod prečo musia byť funkcie pre globálne háky umiestnené v samostatných knižniciach je ten, že každý proces, ktorý bude volať danú funkciu, si ju musí nahrať do svojho pamäťového priestoru, ako môžeme vidieť na obrázku 2.1. Na obrázku je možné vidieť hákovací server, čo je užívateľom vytvorená aplikácia, ktorá bude spravovať háky. Niekoľko inštancií DLL knižnice v pamäťovom priestore každého procesu. Obrázok a znalosti o hákoch boli prevzaté zo zdroja [14], obrázok bol modifikovaný. Čo sú a ako fungujú procesy a dynamicky pripájané knižnice bude vysvetlené v kapitolách 2.4.1 a 2.3.

Nevýhody hákov sa prevažne týkajú globálnych hákov. Dôvod je jednoduchý. Každá



Obrázek 2.1: Činnosť hákov v operačnom systéme

správa, ktorá je odchytená a spracovaná, zaťažuje procesor a tým pádom znižuje výkon celého systému. Samotný Microsoft odporúča použitie globálnych hákov len pre použitie ako ladiaceho nástroja. Okrem zníženia výkonu OS môže dôjsť aj ku konfliktom s ostatnými aplikáciami, ktoré využívajú rovnaký typ globálnych hákov. Jedna z ďalších nevýhod je, pokiaľ existuje chyba vo funkciách, ktoré spracujú vzniknuté a odchytené udalosti, môže spôsobiť veľa nepríjemností. Keďže sa jedná o zavedenie hákov do celého systému, môžu spôsobiť pád systému alebo prinajmenšom bude docielená potreba reštartovať systém aby sa odstránili, vypli háky.

Je známych niekoľko druhov hákov, ktoré sú pre OS Windows dostupné. Napríklad sú tu háky, ktoré umožňujú sledovať činnosť klávesnice a myši. Jedná sa hlavne o to, aký gombík bol stlačený v kombinácií s iným gombíkom, aký znak bol vygenerovaný. Súradnice myši počas jej pohybu, nad akou oblasťou sa myš nachádza. Môžu mať lokálny aj globálny charakter. Ďalší typ hákov súvisí práve s už spomínaným radom správ. Jedná sa o odchytávanie a prípadné filtrovanie správ po opustení radu správ a tesne pred doručením správ funkcii v aplikácii, ktorá ich má spracovať. Ďalšie háky sa týkajú systémových funkcií alebo napríklad aplikácií ako celku. A to odchytávanie správ, ktoré sú zasielané pri vytváraní okien, maximalizovaní, minimalizovaní, zväčšovaní, pohybe oknom, odstránením správy z radu správ o udalosti súvisiace s myšou alebo klávesnicou. Vďaka tomuto háku môžeme ovplyvniť napríklad veľkosť vytváraného okna tým, že upravíme správu, ktorá v sebe nesie informácie o veľkosti okna pri jeho vytváraní. Podrobnosti o každom type háku je možné nájsť na stránkach Microsoftu [1].

2.2.2 Hákovanie v prostredí .net

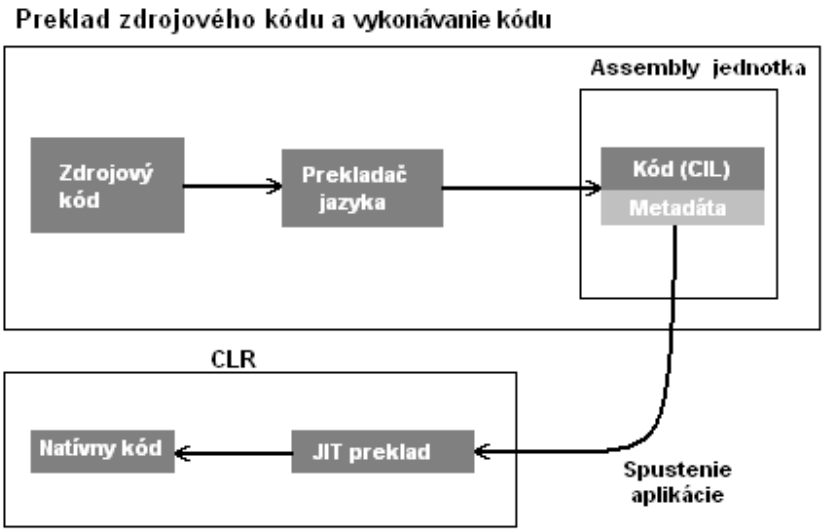
Hákovanie v prostredí .net je mierne odlišné od klasického hákovania pod C++ alebo C. Jedná sa hlavne o štruktúru .net a to, že .net je tzv. *spravovaný* (managed) kód. Aby sme mohli pokračovať s podstatou hákovania v prostredí .net, tak si uvedieme pár základných informácií o prostredí .net.

Prostredie .net je vyvinuté firmou Microsoft. Prostredie .net umožňuje programovanie v rôznych programovacích jazykoch a to z dôvodu existencie prvkov ako je *spoločná infraštruktúra jazyka* (common language infrastructure CLI) [4], *spoločný runtime*[1]*jazyka* (common language runtime CLR) [5] a *spoločný intermediárny jazyk* (common intermediate language CIL) [3]. V podstate sa jedná o to, že .net podporuje viacej programovacích jazykov, ktoré sú optimalizované pre .net prostredie. Je možnosť využiť programovacie jazyky ako je C++, Visual Basic, C# a iné. C# bol jazyk vytvorený Microsoftom, špeciálne pre účely .net prostredia. Každý tento jazyk sa preloží do spoločného intermediárneho jazyka. Preložený kód je obsiahnutý v tzv. *assembly jednotkách* (assembly units). Každá assembly jednotka obsahuje spomínaný CIL a metadáta, ktoré slúžia ako popis CIL kódu. CIL kód sa pri spustení aplikácie preloží. Práve tento preklad spôsobí pomalší štart aplikácií. Aplikácie bežiace na .net sú spúšťané na tzv. *virtuálnom stroji* (virtual machine), ktorý zabezpečí samotný preklad. Tento virtuálny stroj sa pre prostredie .net nazýva CLR. Ukážku prekladu a celého spustenia aplikácie je vidieť na obrázku 2.2. CIL kód môže byť kompilovaný pomocou tzv. *práve včas* (just in time JIT) kompilácie. Aby bolo možné spustiť aplikáciu je potrebné mať nainštalovaný .net framework. Inak sa aplikácia nespustí, pretože CIL nebude možné preložiť a tým pádom ani spustiť. Práve už spomínaný virtuálny stroj spôsobí to, že tento kód sa volá taktiež spravovaný kód. Spravovaním sa myslí hlavne pamäťový priestor vyhradený pre aplikáciu. Hlavnou výhodou .net je rýchlejší vývoj aplikácií. Je to z dôvodu, že .net obsahuje veľké množstvo knižníc s hotovými riešeniami niektorých základných problémov a zabalením systémových funkcií pri vytváraní okien a podobne. V prípade záujmu o danú tému je možné si prečítať obsiahlu knihu [16], ktorá sa zaoberá danou témou.

Natívny, nespravovaný (unmanaged) kód je kód, ktorý je preložený do binárnej podoby. To znamená tento kód je spracovaný priamo procesorom. Hlavnou výhodou je, že program by mal bežať rýchlejšie. Užívateľ si však musí dávať pozor na správu pamäte, čo je značná nevýhoda. Operačný systém využíva nespravovaný kód. Z toho vyplýva, že aj samotné funckie pre zavedenie hákov sú postavené na nespravovanom kóde.

Práve .net samotný spôsobuje problémy, ktoré sa týkajú hákov a ich implementácií v tomto prostredí. Háky zavedené do systému si registrujú funkciu, ktorá ich spracuje. Práve táto funkcia má byť v knižnici DLL. Pokiaľ by bola táto funkcia v spravovanom kóde, spôsobuje to problémy ohľadom adries týchto funkcií. Keďže je .net spravovaný kód a beží na virtuálnom stroji, adresy funkcií vo vnútri aplikácie bežiacej na .net sú vzhľadom na operačný systém neviditeľné alebo lepšie povedané nedostupné štandardným spôsobom. Z čoho vyplývajú problémy. Samotný Microsoft označil globálne háky pre .net ako nepodporované [7]. Jediné dostupné háky pre .net sú pre klávesnicu a myš. Udalosti klávesnice a myši sú totiž udalosti, ktoré po vzniku nepotrebujú, aby im bolo ponechané riadenie. Preto je možné odchytiť tieto udalosti aj pomocou .net čiže spravovaného kódu. Proces spracovania vyzerá takto. Aplikácia má keyboard focus a je aktívna čiže vykonáva svoj kód. Užívateľ stlačí nejakú klávesu. Systém prevezme riadenie od aktívnej aplikácie a predá riadenie

[1]Runtime je časový beh programu. Popisuje priebeh vykonávania programu od jeho začiatku do ukončenia.



Obrázek 2.2: Činnosť prekladača pre prostredie .net

hákovacej aplikácii. To je aplikácia, ktorá zaviedla háky. Následne Windows zavolá funkciu pre spracovanie háku a predá jej správu, ktorá vznikla ako dôsledok vzniknutej klávesovej udalosti. Funkcia to spracuje v pamäťovom priestore procesu hákovacej aplikácie. Windows opäť prevezme riadenie, tentokrát od hákovacej aplikácie. Predá riadenie naspäť pôvodnej aplikácii. V ďalšom kroku pridá vzniknutú správu do radu správ. Aplikácia, v ktorej bola vygenerovaná klávesová udalosť si môže vybrať správu z radu správ a spracovať ju. Obdobné je to aj pre myš. Problém však nastáva pri správach, kedy aplikácia nesmie prísť o riadenie. Jedná sa napríklad o správy typu vytvor okno alebo zmeň jeho veľkosť, polohu. Pri takomto type hákov je postup spracovania nasledovný. Aplikácia je aktívna a vykonáva kód. Následne vytvorí okno. Systém zavolá funkciu pre spracovanie háku. Tentokrát sa však spracovanie odohráva v pamäťovom priestore procesu aplikácie a nie hákovacej aplikácie. To je práve dôvod využitia DLL knižníc a práve preto sa tieto háky nedajú aplikovať priamo na .net prostredie. Informácie použité v tomto odseku boli prevzaté zo zdroja [15].

V prípade, že by bolo účelom vytvoriť aplikáciu v .net, ktorá by spracovala globálne háky by musel byť postup takýto. Pri vzniku udalosti, ktorú hák sledoval sa zavolá funkcia pre spracovanie tohto háku. Funkcia by bola umiestnená v nespravovanom DLL. Vykonávanie by bolo presmerované do .net aplikácie. Čiže do spravovanej aplikácie. Aplikácia by spracovala daný hák. Presmerovala by riadenie naspäť do callback funkcie pre spracovanie háku v DLL. Callback funkcia obdrží informácie ako spracovať danú udalosť z hlavnej aplikácie v .net prostredí. Vykonávanie sa vráti pôvodnej aplikácii, v ktorej vznikla odchytená udalosť.

2.3 Dynamicky pripájané knižníce (DLL)

Dynamicky pripájané knižnice je koncept zdieľaných knižníc. Je využívaný operačným systémom Microsoft Windows a OS/2. DLL knižnice sú rovnaké ako spustiteľné Windows súbory. Môžu obsahovať kód, dáta, zdroje v ľubovoľnej kombinácii. Hlavný dôvod využitia

DLL knižníc bol po vzniku Windows ako viacprocesového operačného systému. Knižnica môže byť súčasne využívaná viacerými procesmi. Knižnica obsahuje hlavnú funkciu, ktorá sa vždy zavolá pri zavedení knižnice danou aplikáciou. Kód existujúcej knižnice je zdieľaný pre všetky procesy. Tento kód vždy zaberá miesto vo fyzickej pamäti PC a nezaberá miesto vo virtuálnej pamäti PC, čiže v stránkovacích súboroch. Ako kontrast ku kódu knižnice sú dáta knižnice. Dáta knižnice sú zvyčajne privátne, využívané len samotným procesom. Každý proces si vytvorí vlastnú kópiu dát do svojho pamäťového priestoru. Každá aplikácia využívajúca takúto DLL knižnicu zavolá systémovú funckiu pre jej zavedenie. Zavedenie znamená, že si aplikácia vytvorí vlastnú inštanciu tejto knižnice a DLL knižnica sa zavedie do pamäťového priestoru aplikácie.

Nevýhody natívnych DLL knižníc spočívajú v tom, že knižnica je závislá na platforme. Čo v praxi znamená, že 32 bitovú knižnicu nie je možné zaviesť do 64 bitového procesu. A tak isto 64 bitovú knižnicu nie je možné zaviesť do 32 bitového procesu. Ďalší problém nastáva pri zdieľaní dát medzi inštanciami knižníc. Keďže každá inštancia knižnice je zavedená do iného pamäťového priestoru, všetky dáta sú inicializované odznova. Problém však vzniká hlavne pri použití ukazovateľov, ktoré nie je možné zdieľať medzi procesmi. Existujú však metódy ako využiť zdieľaný pamäťový priestor operačného systému. Napríklad namapovaním dát, ktoré majú byť zdieľané do tohto priestoru. Problémy súvisiace so zdieľanou časťou pamäte sú v tom, ak užívateľ poškodí dáta v zdieľanej časti pamäte, budú sa všetky procesy využívajúce zdieľanú pamäť chovať nekorektne.

Knižnice môžu byť dynamické alebo statické. Jedná sa pri tom o ich pripájanie k aplikácii, ktorá bude využívať ich funkcie. Knižnice majú väčšinou funkciu ako prostriedok pre zdieľanie rôznych externých funkcií. Statické knižnice sa pripájajú počas prekladu aplikácie. Je to väčšia záťaž na prekladač počas prekladu, pretože musí vyriešiť všetky neznáme adresy a zmeniť ich na pevné adresy. Tento proces býva väčšinou náročnejší ako samotný preklad. Ďalej už nie je potrebná daná knižnica, pretože potrebný kód sa použije pri preklade a je súčasťou výsledného binárneho kódu. Dynamicky pripájané knižnice sú zavedené do programu za behu. Pri preklade aplikácie to nie je takmer žiadna záťaž na prekladač. Kód externých funkcii zostáva v externých súboroch na disku. Väčšina záťaže je presunutá na dobu zavádzania aplikácie alebo počas vykonávania. Samotné pripájanie dynamickej knižnice a zavádzanie externých funkcií je súčasťou operačného systému.

Informácie o dynamických a statických knižniciach boli prevzaté zo stránok wikipedie [6] a [9].

2.4 Procesy, vlákna, moduly

2.4.1 Procesy

Proces (process) je inštancia počítačového programu, ktorý je sekvenčne vykonávaný počítačovým systémom, ktorý má schopnosť konkurentného behu. Program samotný je len sada inštrukcií, pričom proces je aktuálne vykonávanie týchto inštrukcií. Pre jeden program je možné spustiť viacej procesov. Jeden jednojadrový procesor je schopný vykonávať jednu inštrukciu programu v čase, ktoré sú vykonávané jedna za druhou. To znamená jeden proces v čase. Pre beh rôznych procesov súčasne v čase sa využíva časové zdieľanie. V praxi to znamená prepínanie vykonávania medzi rôznymi procesmi. Vždy sa vykonáva iba jeden, kým ostatné procesy čakajú, sú pozastavené. Rýchlosť tohto prepínania je veľmi vysoká, čo vytvára dojem že procesy bežia súčasne. Procesy môžu zdieľať kód programu. Každý proces má však svoju vlastnú inštanciu programu. Preto sa navzájom neovplyvňujú. Pro-

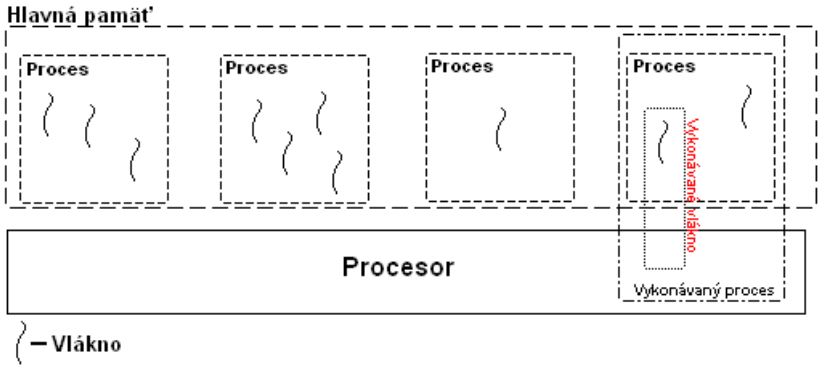
ces po vytvorení získa od operačného systému svoj vlastný pamäťový priestor, v ktorom vykonáva všetky potrebné veci pre svoju činnosť. Ukladá v ňom svoje dáta a ostatné procesy nemajú prístup do tohto priestoru. V prípade, že by mali ostatné procesy prístup do pamäťového priestoru iných procesov, mohli by zmeniť ich dáta čo by spôsobovalo pády aplikácií, v horšom prípade i pád operačného systému. Takto tomu bolo v dobe operačného systému MS-DOS. Medziprocesová komunikácia je umožnená pomocou systémových volaní.

2.4.2 Vlákna

Vlákna (threads) sú veľmi podobné procesom. Vlákno je vykonávaný kód programu. Vlákna sú možnosť ako program rozdeliť na viac súčasne bežiacich úloh. Vlákna a procesy sú odlišné v každom operačnom systéme. Podstatný rozdiel je však v tom, že vlákna sú obsiahnuté vo vnútri procesu a zdieľajú spoločné zdroje, čiže pamäťový priestor, procesy však nie. Technika, viac súčasne bežiacich vlákien, sa nazýva *multithreading*. V rámci jedného procesu môže existovať jedno alebo viac vlákien. Pokiaľ existuje viac vlákien je vykonávané jedno vlákno v čase. Procesor prepína medzi jednotlivými vláknami. Vytvára tak ilúziu súčasného behu vlákien. Ak by počítač obsahoval viac procesorov alebo viacjadrový procesor bolo by možné skutočný súčasný chod rôznych procesov a vlákien. Moderné operačné systémy podporujú súčasný chod aplikácií založený na existencii viacerých procesorov a aj časové prepínanie bežiacich vlákien. Operačný systém umožňuje programátorom manipuláciu s bežiacimi vláknami pomocou systémových volaní. Komunikácia medzi vláknami jedného procesu je bezproblémová, pretože vlákna zdieľajú zdroje a pamäťový priestor. Prepínanie vlákien v rámci jedného procesu je v niektorých prípadoch o mnoho rýchlejšie ako prepínanie samotných procesov. Multithreading je veľmi populárne programovanie aplikácií, ktoré umožňujú súčasný beh viac úloh v rámci jedného procesu. Je potrebné si dávať pozor na správne časovanie týchto bežiacich vlákien z dôvodu možného *uviaznutia* (deadlock). Môže nastať v prípade, že jedno vlákno čaká na dáta od iného vlákna, ktoré čaká dáta od čakajúceho vlákna.

2.4.3 Moduly

Modul je spustiteľný súbor alebo knižnica. Každý proces obsahuje jeden alebo viac modulov. Pre každý modul existuje identifikačné číslo, ktoré je použiteľné pre jeho ovládanie. Prvý modul je spustiteľný súbor.



Obrázek 2.3: Procesy a vlákna v hlavnej pamäti

2.4.4 Zhrnutie

Na záver pre lepšie pochopenie bude ešte zhrnuté čo sú procesy, vlákna a ako spolu súvisia. Proces je najjednoduchšie povedané vykonávaný program. Jedno alebo viac vlákien existuje v rámci jedného procesu. Vlákno je najzákladnejšia jednotka, ktorej operačný systém venuje procesorový čas. Čiže v rámci jedného vykonávaného procesu je vykonávané práve jedno vlákno. Túto situáciu ilustruje obrázok 2.3, na ktorom je možné vidieť hlavnú pamäť, niekoľko procesov a pre každý proces niekoľko vlákien. Je znázornené aj samotné vykonávanie vlákna v rámci vykonávaného procesu. Každý proces má svoj vyhradený virtuálny pamäťový priestor, spustiteľný kód, unikátny identifikátor procesu, prioritu, prístup k systémovým objektom a aspoň jedno vykonávateľné vlákno. Každý proces začne s vykonávaním jedného vlákna a môže vytvoriť viac "súčasne" bežiacich vlákien. Vlákno je entita vo vnútri procesu. Všetky vlákna procesu zdieľajú zdroje a pamäťový priestor procesu. Ako prídavok každé vlákno udržuje informácie o výnimkách, prioritu vlákna, lokálny pamäťový priestor vlákna, unikátny identifikátor vlákna. Informácie o procesoch a vláknach boli získané zo stránok Microsoftu [2] a wikipedie [12], [13]

Kapitola 3

Aplikácia

V nasledujúcej časti si popíšeme návrh aplikácie, knižnice a všetkých potrebných častí. Tak isto budú spomenuté aj problémy, ktoré museli byť riešené a čo ich spôsobilo. Podrobný popis návrhu bude uvedený v kapitole Implementácia 5.

3.1 Súčasný stav

Monitorovanie udalostí operačného systému nepatrí k najpopulárnejším a najrošírenejším témam pre programátorov. Existuje niekoľko druhov aplikácií k monitorovaniu operačného systému Windows, vytvorených pomocou Windows hákov. Niektoré slúžia priamo iba k monitorovaniu správ, niektoré umožňujú aj rozšírenie funkcionality existujúcich aplikácií. S implementáciou globálnych hákov v prostredí .net som sa zatiaľ nestretol. Hlavný dôvod je, že nie sú podporované v prostredí .net.

Cieľom bolo vytvoriť aplikáciu v prostredí .net v kombináci s DLL knižnicou vytvorenou v C++. Umožniť ich vzájomnú komunikáciu a monitorovať niektoré systémové udalosti. Jednalo sa hlavne o monitorovanie udalostí klávesnice a myši a niektoré ďalšie podľa vlastného výberu.

3.2 Návrh aplikácie

Prvý návrh aplikácie bol veľmi jednoduchý. Úmysel bolo vytvoriť aplikáciu, ktorá bude monitorovať iba udalosti klávesnice a myši. Čo je v podstate základné zadanie práce. Zvolený programovací jazyk bol C# a to z niekoľkých dôvodov. Jedná sa v podstate o jeden z najmodernejších programovacích jazykov a bol vyvinutý firmou Microsoft. Aplikácia je určená k monitorovaniu udalostí operačného systému Windows, pochádza taktiež z dielne Microsoftu. Je to čisto objektovo orientovaný jazyk. Hlavné bolo zdokonalenie sa v tomto jazyku, pretože je to jazyk, ktorý má výhľady do budúcnosti.

Nasledovalo nájdenie najvhodnejšieho spôsobu, ktorý by umožňoval globálne monitorovanie udalostí operačného systému Windows. Prvé návrhy boli s využitím slučky správ a ich spracovaním. Ukázalo sa však, že táto metóda funguje iba lokálne pre vlákno bežiacej aplikácie. Neskôr boli nájdené systémové funkcie, ktoré umožňujú hákovanie. Jedná sa o globálnu záležitosť systému a je najvhodnejšia pre takýto druh aplikácie.

Prvý návrh aplikácie bol vytvorený a realizovaný. Nebol to problém a pretože ho bolo možné celý realizovať na platforme .net, čiže priamo v príslušnom jazyku C#. Bol v podstate celkom jednoduchý a tak nasledovalo zvýšenie jeho náročnosti. Jednalo sa o zavedenie

monitorovania pre iné udalosti ako boli udalosti klávesnice a myši. Problém však nastal v prípade práve týchto udalostí, pretože nie sú podporované na platforme .net. A tak bola vytvorená knižnica v jazyku C, ktorá mala za účel vytvoriť komunikačný spoj medzi operačným systémom a aplikáciou napísanou v jazyku C#. S tým vznikli ďalšie problémy, ktoré budú opísané v nasledujúcich sekciách.

Aplikácia samotná bola navrhnutá tak, aby zaznamenávala udalosti do tzv. *logu*, čiže záznamu. Cieľom bolo uchovať základné informácie o vzniknutej udalosti, čiže jej názov. K základným informáciam pridať nejaké rozšírenia. Jednalo sa hlavne o parametre odchytenej udalosti. Ďalšie rozšírenie bolo zobrazenie času, kedy udalosť vznikla a ktorý hák ju odchytil. Keďže aplikácia umožňuje hákovanie globálneho aj lokálneho charakteru, tak informácia o tom, ktorý hák ju odchytil, sa týka práve toho, či sa jednalo o lokálny alebo globálny hák a jeho názov. Niektoré udalosti obsahujú extra informácie o vzniknutej udalosti. Od aplikácie sa očakávalo, aby mohla svoju činnost zastaviť a opätovne spustiť. Umožniť uloženie vytvoreného záznamu udalostí do externého súboru na pevný disk. Boli vytvorené dva návrhy uloženia záznamu na disk. Jeden vo formáte XML[1]súboru a druhý vo formáte textového súboru s príponou log. Vytvoriť nastavenia aplikácie, kde sa budú dať zapínať a vypínať požadované háky. Nastavenia jednotlivých hákov, to znamená vytvoriť filter udalostí (správ) pre jednotlivé háky. Ďalšia časť návrhu bola umožniť aplikácii chod v minimalizovanej forme, v podobe ikonky tzv. *system tray*. Pre lepšiu ovládateľnosť bolo vytvorené kontextové menu pre ikonu v system tray. Účel bolo vložiť do neho základné prvky ovládania aplikácie.

3.3 Návrh knižnice

Návrh knižnice bol vytvorený pre programovací jazyk C++. Dôvod bol, aby knižnica dokázala spracovať globálne háky a dokázala komunikovať s hlavnou aplikáciou napísanou v C#. Návrh bol jednoduchý a to spracovať vzniknutú udalosť a preposlať potrebné informácie do hlavnej aplikácie. Vytvoriť v knižnici funkcie pre zavedenie nových hákov a pre zrušenie existujúcich hákov podľa požiadaviek užívateľa. Z dôvodu vzniknutých problémov ohľadom komunikácie s hlavnou aplikáciou bol navrhnutý systém komunikácie s aplikáciou pomocou správ, o ktoré sa stará samotná knižnica. Bližší popis knižnice bude v sekcii Implementácia 5.

[1]XML je rozšíriteľný značkovací jazyk, určený pre štrukturované uchovávanie dát.

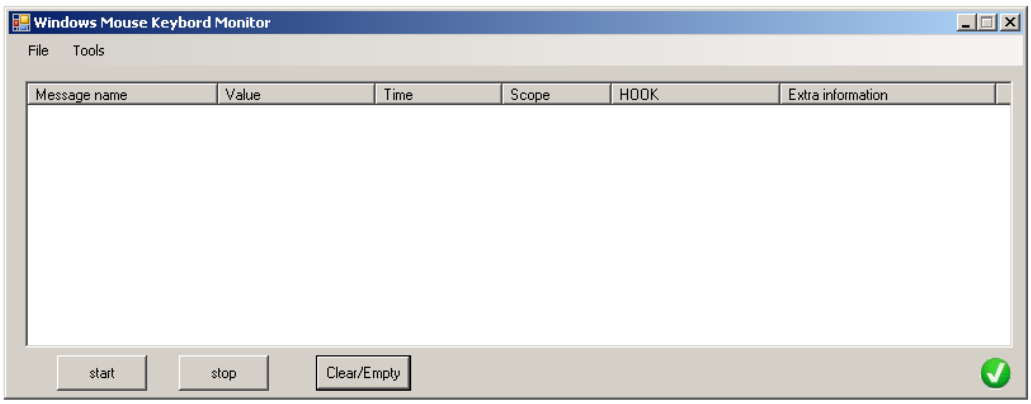
Kapitola 4

Popis prostredia aplikácie

Kapitola sa venuje krátkemu popisu vytvorenej aplikácie a jej prvkov.

4.1 Záznam udalostí

Záznam udalostí je prevedený vo forme listu. Jedná sa o hlavné a taktiež jediné okno, ktoré aplikácia obsahuje. Okno obsahuje menu, tri gombíky, start, stop, clear. Hlavnú časť okna tvorí zoznam zaznamenaných udalostí. Zoznam má 6 stĺpcov name, value, time, scope, hook, extra information. Okno obsahuje ikonku, ktorá hovorí o aktuálnom stave aplikácie. Menu obsahuje dve položky, ktoré sú tiež menu a to konkrétne file a tools. Menu file obsahuje položky pre uloženie záznamu do externých súborov vo formáte log alebo XML, položku pre ukončenie aplikácie. Menu tools obsahuje položku settings, ktorá obsahuje nastavenia aplikácie. Gombík start slúži k spusteniu hákov, ktoré sú v nastaveniach nastavené ako aktívne. Stop slúži k vypnutiu všetkých hákov. Clear vyčistí zoznam zaznamenaných udalostí. Ako to celé vyzerá si môžeme prezrieť na obrázku 4.1.



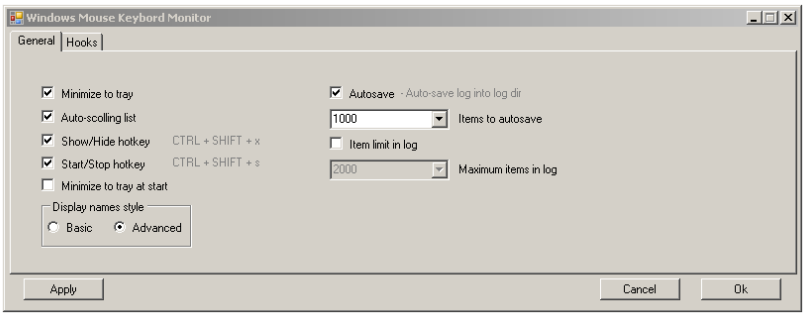
Obrázek 4.1: Hlavné okno aplikácie

4.2 Nastavenia aplikácie

Nastavenia aplikácie je tá najdôležitejšia časť. Jedná sa hlavne o globálne nastavenia aplikácie a o nastavenia hákov samotných. Nastavenia hákov samotných, sú nastavenia, ktoré správy a udalosti sa majú odchytávať, čiže sa jedná o ich filter. Nastavenia obsahujú tri gombíky apply, cancel, ok. Apply slúži pre aplikovanie vykonaných zmien v nastaveniach. Cancel slúži pre návrat do hlavného okna bez uloženia zmien. Ok slúži pre návrat do hlavného okna s uložením zmien. V nastaveniach (settings) sú dve hlavné záložky. Jedna je pre všobecné nastavenia a druhá pre nastavenia hákov.

4.2.1 Všeobecné

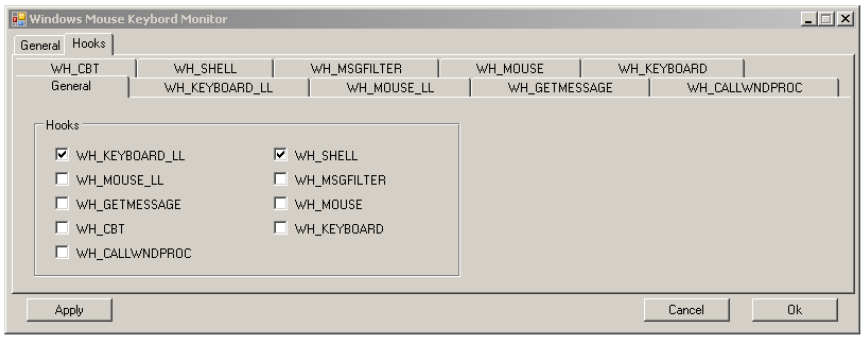
Záložka so všeobecnými nastaveniami (general) obsahuje zaškrtávacie gombíky pre nastavenia aplikácie. Minimize to tray nastavenie umožňuje minimalizáciu do formy ikony v oblasti system tray. Auto-scrolling list zapína a vypína automatické posúvanie sa v zozname záznamov. Znamená to, že po pridaní novej udalosti do zoznamu sa list automaticky posunie na najnižšiu polohu v zozname, aby bolo vidieť novú položku. To v prípade že je táto možnosť zaškrtnutá. Show/hide hotkey umožňuje v použitie globálnej klávesovej skratky pre minimalizovanie do tray ikony a naspäť. Start/stop hotkey umožňuje použitie globálnej klávesovej skratky pre vypínanie a zapínanie hákov. Minimize to tray at start umožnuje nastaviť aby aplikácia po štarte bola automaticky minimalizovaná do tray ikony. Autosave položka nastaví aplikáciu, aby automaticky po určitom počte položiek uložila svoj obsah do externého súboru na disku. Počet položiek je možné nastaviť pomocou zoznamu možných hodnôť umiestneného hneď pod položkou autosave. Súbor sa bude nachádzať v adresári logs jeho názov bude obsahovať dátum a čas kedy bol vytvorený. Bude uložený vo formáte log. Záznam správ sa následne vyčistí. Ďalší a zároveň posledný prvok je pre nastavenie limitu prvkov v zázname. Zoznam položiek pod ním umožnuje nastaviť daný limit. Účelom týchto položiek ako sú tzv. *autoukladanie* a limit položiek v zázname je zvýšenie výkonu aplikácie. Posledné nastavenie dáva užívateľovi možnosť zvoliť spôsob zobrazenia názvov hákov a správ v nastaveniach. Jedná sa o tzv rádio gombík, ktorý má dve možnosti a to základné zobrazenie (basic) a pokročilé (advanced). Pokročilé nastavenie zobrazuje názvy hákov a správ ako sa skutočne volajú v systéme. Základné zobrazenie je jednoduchšie názvy sú zrozumiteľnejšie pre bežných ľudí. Ukážku všeobecných nastavení nájdeme na obrázku 4.2.



Obrázek 4.2: Všeobecné nastavenia aplikácie

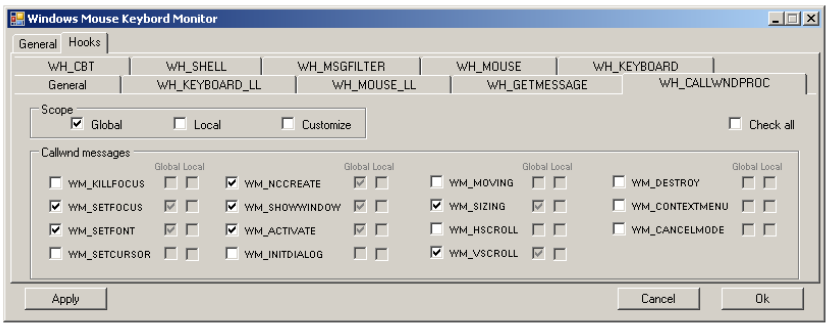
4.2.2 Filter Správ

V záložke háky (hooks) sa nachádzajú nastavenia hákov a odchytávaných správ. Obsahuje v sebe záložky pre jednotlivé háky a jednu záložku so všobecnými nastaveniami pre háky. Záložka so všeobecnými nastaveniami pre háky obsahuje zaškrtávacie gombíky pre všetky háky, ktoré je možné v aplikácii zapnúť. Príklad všeobecných nastavení je možné vidieť na obrázku 4.3 Každá záložka týkajúca sa hákov obsahuje rovnaké typy položiek. Jedná sa



Obrázek 4.3: Nastavenia hákov

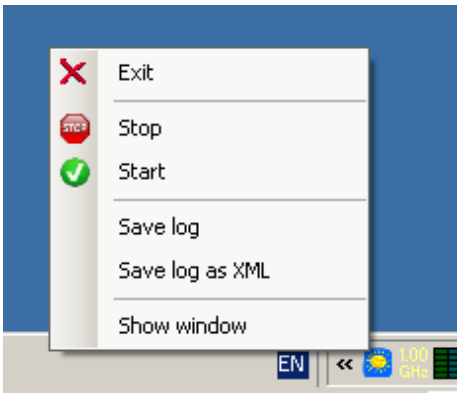
o zaškrtávacie gombíky rozdelené do rôznych zoskupení. Prvé z nich je zoskupenie troch zaškrtávacích gombíkov s názvom scope. Jedná sa o nastavenie hákov a o ich sféru uplatnenia, či sa bude jednať o lokálny alebo globálny hák. Zaškrtnutím global sa nastaví to, že všetky odchytávané správy pre tento hák budú globálneho charakteru. Pre lokálny zaškrtávací gombík platí to isté, ibaže sa bude jednať o lokálny charakter správ. V prípade ak by bol požadovaný špeciálny prípad, keď niektoré správy by boli iba lokálneho a niektoré globálneho charakteru, je tu možnosť zaškrtnúť položku customize. Tá umožní nastavenie jednotlivých správ iba lokálne, alebo globálne, alebo súčasne obidve nastavenia. Ďalšou skupinou zaškrtávacích gombíkov je skupina správ, ktoré môžu byť pre tento hák odchytávané. Ku každej správe sú priradené dva zaškrtávacie gombíky, ktoré hovoria o jej globálnom alebo lokálnom nastavení. Nachádza sa tam ešte jeden zaškrtávací gombík, ktorý po zaškrtnutí spôsobí zaškrtnutie všetkých správ. Názorný príklad globálnych nastavení je možné nájsť na obrázku 4.4.



Obrázek 4.4: Globálne nastavenia pre hák CallWindowProc

4.3 Tray ikona

Posledná dôležitá časť aplikácie je minimalizovaná aplikácia do formy tray ikony. Kliknutím pravého gombíka sa zobrazí kontextové menu, kde zvolíme jednu z možností Exit, Start, Stop, save log, as XML, show window. Jedná sa o tie najzákladnejšie funckie aplikácie. Je to z dôvodu aby mohla byť aplikácia ovládaná aj bez jej plného zobrazenia. Exit prvok slúži k ukončeniu aplikácie. Start a stop slúžia k zapínaniu a vypínaniu bežiacich hákov. Save prvky slúžia k uloženiu záznamu udalostí vo formáte log alebo XML. A prvok show window zobrazí celé okno aplikácie a opustí tray podobu.



Obrázek 4.5: Tray ikona a kontextové menu

Kapitola 5

Implementácia

Aplikácia sa skladá z dvoch hlavných častí, z knižnice DLL a hlavnej aplikácie. Knižnica bola implementovaná v jazyku C++. Konkrétne sa jedná o verziu Visual C++ 9,0 obsiahnutú v .net framework 3,5. Hlavná aplikácia bola implementovaná v C# verzia 2,0. Dôvod implementácie oboch časti v rôznych programovacích jazykoch, bola podpora globálnych hákov v .net, ktorá je pomerne obmedzená. Aby bolo možné preložiť knižnicu aj samotnú aplikáciu súčasne, bol využitý program Visual Studio 2008 od firmy Mircrosoft. Aplikácia slúži k monitorovaniu udalostí OS Windows, ako už bolo spomenuté, za pomoci systémových hákov. Pre správnu činnosť hákov musia byť funkcie pre spracovanie hákov uložené v knižnici DLL. Dôvod je jednoduchý, musí sa jednať o nespravovaný kód, ktorý je za pomoci C++ relatívne jednoduché vytvoriť. DLL napísané v C++ bude mať vystupný formát v binárnom kóde určenom priamo pre procesor. Hlavná aplikácia bola rozvrhnutá do niekoľkých tried. Každá trieda bude opísaná v nasledujúcich sekciách. Prvá trieda, ktorá sa po spustení aplikácie načíta je statická trieda `Program`. Trieda obsahuje iba základné inicializačné nastavenia pre aplikáciu a spustenie, vytvorenie nekonečnej slučky správ pre aplikáciu a vytvorenie inštancie základnej triedy.

5.1 Hlavné okno

Hlavné okno je trieda, ktorá sa zobrazí hneď po spustení aplikácie. Je to základná trieda aplikácie. Názov triedy je `mainform`. Trieda mainform dedí z .net triedy pre formuláre, zvanej `Form`. Okno bolo navrhnuté veľmi jednoducho. Obsahuje len niekoľko základných komponent, určených pre ovládanie a zobrazenie činnosti aplikácie. Hlavnú časť okna tvorí komponenta nazvaná `Listview`. Je to komponenta pre zobrazenie prvkov s rôznym rozložením prvkov a rôznym zobrazením komponent. Je použité zobrazenie detailov, čo v praxi znamená, že celé zobrazenie je rozvrhnuté do niekoľkých stĺpcov. Trieda počas inicializácie vytvorí všetky potrebné triedy, ktoré bude používať. Jedná sa hlavne o triedy pre komunikáciu s nespravovaným DLL a pre triedu spravujúcu háky v .net. Inicializuje konštanty, ktoré sú použité pre spracovanie globálnych hákov. Vytvorí pracovný adresár *logs* pre automatické ukladanie záznamov. Ďalej vytvorí triedu pre prenos globálnych nastavení aplikácie. Zavedie všetky potrebné metódy, ktoré budú slúžiť ako tzv. *Event Handlery*. Sú to metódy pre spracovanie udalostí, ktoré slúžia pre komunikáciu so zbytkom aplikácie. Trieda obsahuje dve metódy pre pridávanie do listview. Prvá slúži k pridaniu štandardného riadku do komponenty Listview, druhá je len pre rozšírenie existujúceho riadku o posledný možný stĺpec. Ďalšie dve metódy slúžia pre zavedenie a spracovanie systémových klávesových skratiek.

Trieda obsahuje metódy pre správu veľkostí okna a pre obsluhu ostatných komponent, ako sú gombíky a menu.

5.2 Háky a ich implementácia v .net

Kompletnú správu hákov má na starosti trieda `windows_hooks`. Jedná sa o spracovanie globálnych hákov v prostredí .net, spracovanie všetkých lokálnych hákov, zavedenie a zrušenie všetkých hákov, čiže ich inštalovanie a odinštalovanie. Taktiež obsahuje deklarácie externých systémových funkcií využitých v celej aplikácii. Základ hákovania tvoria metódy `install_hooks` a `uninstall_hooks`. ich parameter je jedno jediné 32 bitové číslo, inak nazývané aj *dvojslovo* (DoubleWord). Toto číslo má nastavené príslušné bity podľa hákov. Každý bit odpovedá jednému háku. Spodné slovo dvojslova tzv. *low order word* odpovedá globálnym hákom a horné slovo dvojslova tzv. *high order word* odpovedá lokálnym hákom. Metóda pre zavedenie hákov získa hodnoty príslušných aktívnych hákov za pomoci logickej funkcie AND. Aktívne háky majú nastavenú logickú hodnotu na 1. Obdobne to funguje aj pre metódu zrušenia hákov. Háky, ktoré majú byť zrušené, majú logickú hodnotu taktiež 1. Jedná sa o dve rôzne čísla. Jedno pre aktívne háky a druhé pre neaktívne háky. Pre zavedenie globálnych hákov nepodporovaných v .net sa volá priamo funkcia knižnice pre zavedenie hákov. Pre zavedenie lokálnych hákov a globálnych hákov podporovaných v .net je využitá metóda `install_hook_managed`. Keďže sa v tejto metóde využívajú systémové funkcie, jedná sa o komunikáciu priamo s nespravovaným kódom. Funkcii pre zavedenie háku sa musí predať ukazovateľ na funkciu, ktorá spracuje odchytenú udalosť. Pre toto použitie sa hodia delegáti prostredia .net, ktorí sú istá forma náhrady ukazovateľov na funkcie. Trieda ešte obsahuje metódy, ktoré sú vyžadované pri spracovaní hákov v príslušných metódach. Každá metóda pre spracovanie hákov má základné funkcie a to v prvom rade overiť, či môže danú udalosť spracovať. Následne spracuje parametre udalosti a výsledok odošle tak, že vytvorí udalosť, ktorú spracuje trieda mainform.

5.3 Windows konštanty

Konštanty pre aplikáciu sú uložené pomocou *výčtového typu* `Enum` – enumerácií a pomocou jednej triedy. Enumerácie obsahujú všetky konštanty prepísané podľa Windows konštánt. Dôvod tohto relatívne zbytočného prepisovania bol, že .net neobsahuje žiadne konštanty týkajúce sa operačného systému Windows. Taktiež sú vytvorené konštanty v podobe enumerácií pre účely aplikácie. Trieda pre konštanty je veľmi jednoduchá. Táto trieda sa nazýva `windows_hook_constants`. Obsahuje dve hlavné statické metódy. Metóda `set_up_messages` načíta identifikačné čísla správ z knižnice DLL. Jedná sa o užívateľské správy registrované v systéme za pomoci funkcií v knižnici. Sú určené pre komunikáciu s hlavnou aplikáciou. Identifikačné čísla správ sa priradia odpovedajúcim premenným v triede. Ďalšia metóda slúži k vytvoreniu tzv. *slovníka* (Dictionary). Metóda sa volá iba raz pri inicializácii hlavnej triedy mainform. Metóda sa volá `init_constants`. Je to slovník, ktorý slúži na prevod číselných konštánt OS Windows na odpovedajúce reťazcové (slovné) konštanty. Taktiež je tu využitie i pre iné preklady určené pre aplikáciu samotnú. V triede sa nachádza ešte niekoľko metód určených pre získanie príslušných slovníkových konštánt. Je to najmä niekoľkokrát preťažená metóda `translate_constant`.

5.4 Štruktúry

Pre korektnú činnosť aplikácie bolo okrem konštánt operačného systému Windows vyžadované vytvoriť štruktúry, ktoré by presne kopírovali existujúce systémové štruktúry, ktoré systém využíva pre hákovanie. Zároveň bolo potrebné vytvoriť štruktúry, ktoré by boli určené pre účely aplikácie a triedu pre uchovávanie a spracovanie nastavení aplikácie. Globálne nastavenia aplikácie je možné v prostredí .net ukladať za pomoci statickej triedy `Settings`, ktorá je súčasťou každej aplikácie vytvorenej v .net pomocou Visual Studio 2008. Je to trieda, ktorá umožňuje uchovávať nastavenia aplikácie do súborov pre daného užívateľa alebo pre celú aplikáciu. Práve túto možnosť využíva aj trieda `application_settings`, ktorá slúži pre vnútorný prenos nastavení za behu aplikácie. Obsahuje metódy pre uloženie dát do samotnej inštancie triedy a súčasne do statickej triedy `Settings`, ktoré sú ukladané do externých súborov a umožňujú uchovávať informácie o nastaveniach aplikácie aj po vypnutí aplikácie. Trieda má dva konštruktory. Jeden pre vytvorenie prázdnej inštancie triedy a druhý pre načítanie dát zo statickej triedy `Settigns`. Štruktúry vytvorené priamo pre aplikáciu sú dve štruktúry. A to menovite štruktúra `msg_data` pre prenos informácií o prijatej správe medzi triedou spracujúcou správu a triedou zaznamenávajúcou správu. Druhá štruktúra `msg_flags` uchováva príznaky prijatej správy. Zvyšné štruktúry sa týkajú parametrov prijatých správ. Jedná sa o to, že systém používa na prenos dát len dva parametre. Preto pri zložitejších správach sú použité práve štruktúry na ich prenos. Aby bola dosiahnutá kompatibilita so systémovými štruktúrami sú vytvorené ich presné kópie. Aplikácia prenesie obsah štruktúry z parametru správy do lokálnej štruktúry. Príklad týchto štruktúr je štruktúra `rectangle`, ktorá je kópiou systémovej štruktúry `RECT` slúžiacej pre uchovanie dát o súradniciach okna, ktoré tvorí obĺžnik. Alebo je to systémová štruktúra `mouse_hook_struct`, ktorá nesie informácie o vzniknutej udalosti vytvorenej vstupom z myši. Ostatné existujúce štruktúry sú taktiež kópiami existujúcich systémových štruktúr určených hlavne pre hákovanie.

5.5 Knižnica DLL pre globálne háky

Knižnica je nezávislý súbor DLL od aplikácie. DLL knižnica sa skladá z dvoch základných častí a to zavádzacia funkcia taktiež známa ako *hlavná funkcia DLL* (DLLMain) a hlavná časť knižnice, ktorá obsahuje funkcie, ktoré knižnica implementuje. Druhú časť môžme tiež nazvať jadro knižnice.

5.5.1 Zavedenie knižnice

Zavedenie knižnice sa vykonáva pomocou funkcie main. Funckia je volaná pri zavádzaní aj pri rušení inštancie knižnice do procesu alebo vlákna, ktoré ju vyvolalo. V rámci zavádzania knižnice sa musia vždy vykonať všetky operácie pre jej správny chod. V tomto prípade sa jednalo hlavne o registrovanie vlastných správ do systému. Registrácia správ znamená vyhradenie identifikačného čísla pre správu, čím sa zaručí, že žiadna iná správa v operačnom systéme nebude mať rovnaké identifikačné číslo. O vytvorenie a registrovanie všetkých správ sa stará funcia `init_messages` a ich hodnoty naplní do poľa celočíselných hodnôt.

5.5.2 Jadro knižnice

Hlavná časť knižnice je tvorená funkciami pre obsluhu vzniknutých udalostí, ktoré sú registrované do operačného systému pomocou systémových funkcií. Existuje funkcia, ktorá po zavolaní vráti ukazovateľ už na existujúce pole správ. Funkcia je určená pre aplikáciu, ktorá môže využívať danú DLL aby mohli spolu komunikovať. Obdobne ako pri hákoch podporovaných priamo v .net aj tu existujú funkcie pre zavedenie globálnych hákov do systému. Konkrétne sa jedná práve o funkcie, ktoré sú volané z už spomínaných metód pre zavedenie a zrušenie hákov všeobecne. Funkcia `install` slúži k zavedeniu hákov do systému. Aby sa predišlo zavedeniu už existujúceho háku prevádza sa kontrola na existenciu háku a kontrola parametru podľa ktorého sa určí, ktorý hák má byť zavedený. Pre zrušenie už existujúceho háku sa použije funkcia `uninstall`, ktorá taktiež prevádza kontrolu na existenciu háku, aby sa predišlo zbytočným volaniam funkcií. Ostatné funkcie sú všetky typu tzv. *hookproc*. Jedná sa o typ funkcií taktiež nazývaných hákovacie procedúry. Podobný typ metód už bol spomínaný pri spracovaní hákov podporovaných priamo v .net. Boli to metódy pre spracovanie vzniknutých udalostí. Tieto hookproc funkcie sú prakticky identické. Funkcie najskôr spracujú parameter, ktorý hovorí o tom, či môže byť zachytená udalosť spracovaná. Ak áno nasleduje výber užitočných informácií z parametrov funkcie. V prípade, že sa narazí na správu, ktorá má byť spracovaná aj v rámci aplikácie, správa sa pošle hlavnej aplikácii. Aby nedošlo ku konfliktu pri spracovaní správy a aby aplikácia vedela, že správa ktorá bola doručená, je správa určená len pre záznam a nie je to správa od systému, preto sú použité práve už spomínané vlastné správy. Aby bolo zjednodušené použitie vlastných správ, tak je zavedená konvencia pre ich názvy. Správa má identický názov so systémovou správou s rozdielom, že na začiatku názvu správy je pridané písmeno m (ako moja). Všetky ostatné funkcie implementované v knižnici sú práve typu hookproc. Jedná sa o funkcie fungujúce na rovnakom princípe ako práve uvedený príklad. Knižnica implementuje háky typu Get Message, CBT, Call Window Proc, Message Filter, Shell. Pre bližšie informácie o konrétnych hákoch je potrebné navštíviť stránky Microsoftu. [1].

5.5.3 Vzniknuté problémy a ich riešenie

Pri implementácii knižnice vzniklo nadmerné množstvo neočakávaných problémov. Jeden z najväčších problémov bola komunikácia medzi aplikáciou a knižnicou. Prvé riešenie bolo zaviesť do knižnice ukazovateľ na funkciu z hlavnej aplikácie, ktorá by sa volala vždy pri vzniknutej udalosti. Spracovanie by bolo v hlavnej aplikácii. Toto fungovalo iba pokiaľ bola aplikácia aktívna. Dôvod je jednoduchý, keďže sa jedná o knižnicu DLL, ktorá sa vždy zavedie do pamäťového priestoru procesu, ktorý ju vyvolal, tak bola zťažená komunikácia. Jednalo sa totiž o medzi procesovú komunikáciu, ktorá je veľmi zložitá. Nebola to však obyčajná medziprocesová komunikácia, pretože navyše k tomu to bola komunikácia medzi spravovaným a nespravovaným kódom. Ďalšie riešenie bolo za pomoci vlastných správ. Jednalo sa o prvé riešenie, ktoré bolo funkčné aj pokiaľ aplikácia nebola aktívna. Správa môže mať dva parametre. Pre prenos zložitejších dát boli využité ukazovatele na štruktúry. Tie však strácali zmysel pokiaľ sa jednalo o správu od iného procesu, pretože ukazovali do pamäťového priestoru iného procesu. Nasledujúci pokus bol neúspešný. Jednalo sa o prenos ukazovateľov medzi procesmi, ktorý sa však ukázal ako nefunkčný v kombinácii s prostredím .net. Posledné, funkčné a zároveň používané riešenie je pomocou niekoľkých správ odoslaných za sebou. Podstata je, že pomocou správ je možné prenášať celočíselné hodnoty o veľkosti 32 bitov, ktoré sa prenesú korektne aj v prípade, že správu odoslal iný proces. Jedná sa totiž o posielanie dát samotných a nie ukazovateľa na dáta. Prvá správa

obsahuje v sebe názov pôvodnej správy a nejaké dve hodnoty, ktoré budú spracované v aplikácii. Ak je potrebné preniesť viac ako dve hodnoty použijú sa ďalšie správy. Názov správy hovorí o tom, že sa jedná o rozšírenie predchádzajúcej správy. Parametre sú hodnoty, ktoré sú požadované. Aplikácia je následne pripravená pre ich spracovanie.

Jeden z existujúcich problémov bol zdieľanie dát medzi samotnými inštanciami knižnice. Každá DLL knižnica obsahuje nejaké dáta ako každý iný kód, ktoré slúžia k jej správnej činnosti. Aby každá inštancia knižnice fungovala korektne, je potrebné aby zdieľali dáta. K tomu slúži zdieľaný úsek pamäti, ktorý je poskytnutý operačným systémom.

5.6 Spracovanie správ z DLL

Ako už bolo povedané komunikácia medzi knižnicou a aplikáciou prebieha pomocou vlastných správ. Správy sú v aplikácii spracované v slučke správ, ktorá rozhoduje o tom, či má byť správa odoslaná vnútorným častiam aplikácie pre ďalšie spracovanie alebo nie. Aby bolo možné umožniť užívateľovi vlastné spracovanie správ, má k dispozícii triedu, ktorá keď implementuje rozhranie `IMessageFilter`, tak je možné ju použiť pre filtrovanie správ celej aplikácie. Táto trieda je zavedená priamo do aplikácie pomocou, statickej metódy `AddMessageFilter`. Cez tento tzv. filter správ prechádzajú všetky správy určené aplikácii. Je možné rozhodnúť o tom, či bude správa predaná aplikácii alebo nie. V prípade tejto aplikácie postačovalo, aby boli správy prechádzajúce týmto filtrom monitorované. Implementovaná trieda má názov `msg_filter`. Obsahuje metódu, ktorá vykonáva samotné filtrovanie. V tomto prípade sa jedná iba o monitorovanie. Keďže sa jedná o neštandardné správy, je to najvhodnejší spôsob monitorovania a samotného spracovania správ. Metóda testuje, či je prijatá správa jedna z vlastných správ. Ak áno, tak sa naplní štruktúra `msg_data` hodnotami, ako sú názov správy a hodnota správy. Hodnota správy sa získa z parametrov prijatej správy. Vytvorí sa nová udalosť a mainform trieda ju zpracuje. Pokiaľ by sa jednalo o správu pre ktorú by boli odoslané ďalšie rozširujúce informácie, aplikácia ich musí spracovať korektne. Filter správ po prijatí rozširúcej správy zistí o koľkú rozširujúcu správu sa jedná. Ak by bola prijatá posledná rozširujúca správa, aplikácia nastaví názov správy pre rozširujúce informácie podľa poslednej prijatej správy. Následne ich odošle pomocou udalosti aplikácie a trieda mainform ich správne spracuje. Ak by nebola prijatá posledná rozširujúca správa, aplikácia nastaví príznak v štruktúre `msg_flags`, ktorý hovorí o poradí správy.

V triede sú implementované metódy, ktoré sú využité pri spracovaní parametrov správ. Tieto metódy využívajú hlavne systémové funkcie, ako napríklad funkcie pre zistenie názvu okna, názvu triedy okna, názvu koreňového okna a funkcie pre získanie horného a dolného slova z dvojslova.

5.7 Okno pre nastavenia aplikácie a hákov

Do okna obsahujúceho nastavenia aplikácie je možné sa dostať pomocou menu v hlavnom okne, kliknutím na položku *Tools* a následne na položku *Settings*. Nastavenia aplikácie tvoria dve triedy. Obe triedy sú komponenty užívateľsky vytvorené. Dedia z triedy `UserControl`. Prvá a zároveň hlavná trieda pre nastavenia je pomenovaná `settings_control`. Táto trieda obsahuje prvky pre nastavenia aplikácie ako celku a mimo iné obsahuje aj druhú triedu pre nastavenia hákov. Táto trieda sa nazýva `hooks_control_hooks`.

5.7.1 Nastavenia aplikácie

Trieda pre globálne nastavenia aplikácie a pre komunikáciu s triedou pre nastavenia hákov, obsahuje len niekoľko komponent a je celkom jednoduchá. Hlavná komponenta tejto triedy, ktorá je tvorená dvomi záložkami a 3 gombíkmi pre uloženie nastavení, zrušenie nastavení a návrat do hlavného okna a gombík pre uloženie nastavení a návrat do hlavného okna tvoria základ tejto triedy. Záložka pre všeobecné nastavenia a záložka obsahujúca triedu pre nastevenia hákov. Všeobecné nastavenia obsahujú niekoľko zaškrtávacích gombíkov a jeden tzv. rádio gombík. Trieda obsahuje metódy pre nastavenie východiskových nastavení aplikácie po spustení. Obsahuje vlastnú inštanciu triedy `application_settings`, v ktorej uchováva nastavenia aplikácie v rámci tejto triedy. Obsahuje metódy pre správu udalostí v rámci tejto triedy. Ďalej obsahuje metódu pre uloženie nastavení aplikácie do už spomenutej triedy `application_settings`. Táto metóda pozbiera informácie so všetkých komponent obsiahnutých v tejto triede a zavolá metódu triedy `hooks_control_hooks` pre získanie nastavení hákov.

5.7.2 Nastavenia hákov

Nastavenia hákov sú obsiahnuté v tejto triede pomerne komplexným spôsobom. Je to hlavná trieda nastavení, pretože obsahuje úplnú manipuláciu s hákmi. Nastavenia hákov sa týkajú zapínania a vypínania hákov, nastavenia filtrov pre jednotlivé správy v rámci daného háku, nastavenie globálneho alebo lokálneho odchytávania správ. Nastavenia správ pre jednotlivé háky sú uložené v niekoľkých 32 bitových číslach. Pre každý hák jedno číslo. Jednotlivé bity čísel odpovedajú jednotlivým správam. Pre správy globálnych hákov sú to bity, dolné bity dvojslova a pre lokálne háky sú to horné bity dvojslova. Pomocou týchto čísel sa uskutočňuje aj filtrovanie správ v ostatných triedach. Preto sú tieto čísla statické a je k nim prístup iba pre získavanie dát. Modifikácia mimo objektu tejto triedy nie je možná. Hlavnú časť triedy tvorí komponenta tvorená záložkami. Jedna záložka pre všeobecné nastavenia hákov a zvyšné pre nastavenia hákov. Jedna záložka pre jeden typ háku. Každá záložka pre hák obsahuje niekoľko zaškrtávacích gombíkov. Sú rozdelené do skupín. Jedna skupina zaškrtávacích gombíkov je určená pre nastavenie globálnych a lokálnych hákov. Ďalšia skupina zaškrtávacích gombíkov je určená pre filtrovanie správ. Každá správa pozostáva z troch zaškrtávacích gombíkov. Prvý povoľuje odchytávať danú správu, druhý nastavuje, či sa bude jednať o globálny hák a tretí nastavuje, či sa bude jednať o lokálny hák. Pre správnu činnosť týchto hákov sú vytvorené metódy, ktoré zaškrtávajú správy a ich príslušné zaškrtávacie gombíky podľa nastavení prvej skupiny zaškrtávacích gombíkov. V každej záložke existuje ešte jeden zaškrtávací gombík pre rýchle nastavenie všetkých správ.

Trieda obsahuje dve metódy pre komunikáciu s nadradenou triedou. Jedna slúži k nastaveniu všetkých zaškrtávacích gombíkov. Druhá slúži pre získanie dát podľa nastavených zaškrtávacích gombíkov. Ostatné metódy sú určené pre správu nastavení zaškrtávacích gombíkov. Kvôli nadmernému počtu zaškrtávacích gombíkov a lepšej efektivite boli umiestnené do poľa. Nastavenia hákov sa ukladajú do triedy pre globálne nastavenia aplikácie `Settings`.

5.8 Ukladanie záznamu

Ukladanie záznamov je riešené jednoducho pomocou statickej triedy `log_save`. Trieda obsahuje metódy pre ukladanie do súborov v dvoch formátoch. Dve metódy sú určené pre ukladanie vo formáte XML. Jedna metóda spracuje dáta do formátu XML a druhá metóda uloží tieto predspracované dáta na požadované miesto na disku zadané parametrom. Ďalšie dve metódy sú určené pre uloženie záznamu vo formáte log, čo je obyčajný textový súbor. Obe metódy určené pre ukladanie vo formáte log pracujú obdobným spôsobom ako pre formát XML.

5.9 Minimálne požiadavky

Aplikácia je určená špeciálne pre monitorovanie operačného systému Windows. To znamená, že je kompatibilná iba s Windows a v inom operačnom systéme fungovať nebude. Je vyžadovaná minimálna verzia operačného systému Windows 2000 alebo Windows XP. Operačný systém musí byť 32 bitový aby bola docielená kompatibilita s natívnou knižnicou DLL. V operačnom systéme musí byť nainštalovaný minimálne Microsoft .net framework 3.5, ktorý obsahuje Visual C++ 9.0. Aplikácia nie je platformovo nezávislá. Nie sú vyžadované zvláštne hardwarové nároky.

Kapitola 6

Záver

Počas vytvárania práce som musel naštudovať rôzne materiály, keďže sa jedná o pomerne netradičnú tému, zdroj všetkých mojich zdrojov bol internet. Jednalo sa hlavne o hákovanie, o ktorom som nikdy predtým nepočul. Háky, ktoré nachádzajú uplatnenie, nie len pre monitorovanie udalostí OS, ale taktiež poskytujú rôzne prostriedky pre prácu s aplikáciami, ako je ladenie a rozširovanie funkcionality existujúcich aplikácií. Háky sú práve v tomto jedinečné.

Mojou snahou bolo vytvoriť jednoduchú a použiteľnú aplikáciu v rámci možností a schopností. Zaujímavú možnosť, ktorú aplikácia ponúka je presne zvoliť správy, ktoré majú byť sledované a nastaviť ich filter pre lokálny alebo globálny hák. Ďalšie praktické vylepšenie umožňuje registrovať aplikácii systémové klávesové skratky, ktoré by mali zrýchliť prístup k aplikácii a jej ovládaniu. Taktiež existencia ikony aplikácie vo forme systémovej tray ikony, by mala byť istým zlepšením ako monitorovať systém na pozadí.

Práca bola pre mňa veľkým prínosom, čo sa týka programovania i teoretických znalostí ohľadom operačného systému Widnows. Zoznámil som sa s detailným fungovaním procesov, medziprocesovou komunikáciou a s DLL knižnicami. Jednalo sa o prvý väčší projekt, ktorý bol objektovo orientovaný, čím som získal viac skúseností ohľadom objektovo orientovaného programovania a pochopil lepšie niektoré koncepcie OOP v praxi. Taktiež som sa naučil vytvoriť DLL knižnicu, ako s ňou korektne pracovať a hlavne som sa zlepšil v programovaní v programovacom jazyku C#.

Literatura

[1] About Hooks. [online], naposledy navštívené Máj 2008.
URL http://msdn.microsoft.com/en-us/library/ms644959.aspx

[2] About Processes and Threads. [online], naposledy navštívené Máj 2008.
URL http://msdn.microsoft.com/en-us/library/ms681917(VS.85).aspx

[3] Common Intermediate Language. [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Common_Intermediate_Language

[4] Common Language Infrastructure. [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Common_Language_Infrastructure

[5] Common Language Runtime. [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Common_Language_Runtime

[6] Dynamic-link library. [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Dynamic-link_library

[7] Global hooks are not supported in the .NET Frameworks. [online], naposledy navštívené Máj 2008.
URL http://support.microsoft.com/kb/318804

[8] Human interface device. [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Human_interface_device

[9] Library (computing). [online], naposledy navštívené Máj 2008.
URL
http://en.wikipedia.org/wiki/Library_%28computing%29#Dynamic_linking

[10] Message loop in Microsoft Windows. [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Message_loop_in_Microsoft_Windows

[11] Message queue. [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Message_queue

[12] Process (computing). [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Process_%28computing%29

[13] Thread (computer science). [online], naposledy navštívené Máj 2008.
URL http://en.wikipedia.org/wiki/Thread_%28computing%29

[14] IVANOV, I.: Api hooking revealed. [online], december 2002.
URL http://www.codeguru.com/cpp/w-p/system/misc/article.php/c5667/

[15] KENNEDY, M.: Global System Hooks in .NET. [online], január 2005, naposledy navštívené Máj 2008.
URL http://www.codeproject.com/KB/system/globalsystemhook.aspx

[16] TROELSEN, A.: *C# a .NET 2.0 Profesionálně*. Brno: Zoner Press, první vydání, 2006, ISBN 80-86815-42-0, překlad: POKORNÝ, J.