VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

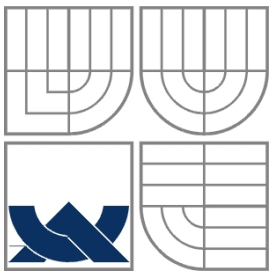
NÁSTROJ PRO MONITOROVÁNÍ SYSTÉMOVÝCH PROSTŘEDKŮ OS WINDOWS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

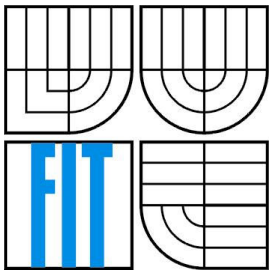
AUTOR PRÁCE
AUTHOR

RADOVAN SYNEK

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NÁSTROJ PRO MONITOROVÁNÍ SYSTÉMOVÝCH PROSTŘEDKŮ OS WINDOWS

APPLICATION FOR MONITORING SYSTEM RESOURCES OF OS WINDOWS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

RADOVAN SYNEK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. LUKÁŠ GRULICH

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Synek Radovan**
Obor: Informační technologie
Téma: **Nástroj pro monitorování systémových prostředků OS Windows**
Kategorie: Operační systémy

Pokyny:

1. Analyzujte požadavky na tento nástroj. Cílem je vytvoření aplikace, která bude v reálném zpracovávat a zobrazovat obsazení systémových prostředků (pamětí, atd.) a informace o procesech
2. Seznamte se s WinAPI, konkrétně s částmi souvisejícími se zadáním. Zvolte vhodný nástroj pro implementaci.
3. Aplikaci implementujte, testujte. Vytvořte uživatelský manuál.
4. Diskutujte možná zdokonalení.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Grulich Lukáš, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Radovan Synek**
Id studenta: 84134
Bytem: Kosinova 283, 533 41 Lázně Bohdaneč
Narozen: 03. 07. 1985, Pardubice
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Nástroj pro monitorování systémových prostředků OS Windows
Vedoucí/školitel VŠKP: Grulich Lukáš, Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☒ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Práce pojednává o koncepci monitorování prostředků v operačních systémech Microsoft Windows, poskytovaných nástrojích, používaných metodách a jejich programovacím rozhraní. Dále obsahuje popis návrhu a implementace vytvořeného nástroje pro monitorování prostředků. Není opomenuto ani testování a zhodnocení aplikace.

Klíčová slova

Monitorování, Windows API, výkon, čítač, Windows, prostředky, procesy, logování.

Abstract

This work deals with conception of monitoring system resources in Microsoft Windows operating systems, granted tools, used methods and it's programming interface. Next, it contains description of design and implementation own created tool for monitoring system resources. Description of testing and valuation of application also hasn't been forgotten.

Keywords

Monitoring, Windows API, performance, counter, Windows, resources, processes, logging.

Citace

Radovan Synek: Monitorování systémových prostředků OS Windows, bakalářská práce, Brno, FIT VUT v Brně, 2007

Nástroj pro monitorování systémových prostředků OS Windows

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Lukáše Grulicha. Další informace mi poskytli Lukáš Polok, Jan Liška a Ing. Michal Vosmanský. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radovan Synek
8.5.2007

Poděkování

Tímto chci poděkovat Ing. Grulichovi za poskytování informací o náležitostech bakalářské práce, dále Lukáši Polokovi a Janu Liškovi za úvod do programování uživatelského rozhraní ve Windows API a Ing. Vosmanskému za poznámky k určení názvu síťového adaptéru.

Obsah

Úvod

Monitorování systémových prostředků a výkonu celého operačního systému nejen neodmyslitelně patří k hlavním úkonům správy serverů, ale má svůj význam i u klientských stanic. První kapitola nazvaná Zásady a postupy monitorování prostředků systému Windows podává přehled o koncepci sledování výkonu operačních systémů Windows a seznamuje s nástroji, které jsou uživateli k dispozici. Druhá kapitola se věnuje možným přístupům, jak vytvořit další podobný nástroj. Zmíněny jsou tři různé alternativy a blíže rozebrána ta z nich, kterou jsem zvolil k řešení. Třetí kapitola obsahuje popis návrhu vlastní aplikace, použité postupy, vývojové nástroje a logické členění programu. Kapitola čtvrtá navazuje na předchozí kapitolu popisem implementace aplikace. Zvláštní pozornost je věnována problémům, se kterými jsem se během vývoje aplikace setkal, a také uživatelskému rozhraní. Pátá kapitola se zabývá testováním aplikace. V závěru je shrnuto toto téma a vlastnosti vytvořeného nástroje, jeho nedostatky a návrhy na vylepšení.

1 Zásady a postupy monitorování prostředků systému Windows

Aby monitorování mělo smysl, je potřeba stanovit tzv. základní linku – přijatelný výkon při typických provozních podmínkách. Tato základní linka je potom referenčním bodem při pozdějším zkoumání výkonu systému. Používání systémových prostředků popisuje propustnost, fronta a doba odezvy.

Propustnost udává množství vykonané práce za jednotku času. Stoupá se zatížením až do maxima, potom vzniká fronta (viz dále). Propustnost celého systému je dána propustností jeho nejpomalejšího prvku (tzv. bottleneck). Fronta je počet úloh čekajících na zpracování, kdy je propustnost systému na maximu a požadavky se nestíhají vyřizovat. Doba odezvy označuje množství času potřebného na provedení určité úlohy.

Nejprve je třeba se seznámit se sledovanými objekty a nástroji, které jsou v systému Windows k dispozici.

1.1 Sledované objekty

1.1.1 Koncepce objektů výkonu

Sledovat lze poměrně širokou škálu objektů, které odpovídají určitému hardwaru, aplikaci nebo službě. Jako příklad objektů sledování výkonu aplikací a služeb to mohou být relace terminálové služby, replikace SQL serveru nebo kompilátor JIT prostředí .NET. To jsou spíše speciální příklady, obecně je potřeba pro určení výkonu počítače sledovat tyto objekty:

- Procesor
- Operační paměť
- Disky
- Síťová rozhraní

Otázka je, jaké parametry lze u těchto objektů sledovat a které z nich budou pro určení výkonu nejvhodnější. Každý objekt má jiné parametry – tzv. čítače výkonu. Konkrétní čítač výkonu je určen řetězcem: `\\jméno_počítače\objekt(instance)\čítač`, kde `jméno_počítače` slouží k určení počítače, který chceme sledovat vzdáleně a v případě, že se zajímáme o lokální počítač, nemusí být zadáno. `Objekt` je objekt reprezentující určitý typ zařízení – např. logický disk a `instance` označuje jednotlivá zařízení, která pod tento objekt spadají – pro logický disk je

instancí například disk C:. Čítač je potom údaj, který chceme o tomto objektu zjistit, např. střední fronta disku nebo počet zápisů na disk za sekundu. Data o výkonu jsou sbírána periodicky.

1.1.2 Čítače

Existují čítače okamžité, které poskytují poslední změřenou hodnotu a čítače průměrující. Průměrujícím čítačem je např. % času procesoru nebo chyby stránek/s. Takové čítače dávají průměrnou hodnotu mezi dvěma posledními měřeními.

Sledovat veškeré čítače výkonu daného objektu je většinou nepraktické, ať už z důvodu zbytečně velkého objemu informací, ale především kvůli zátěži systému. Monitorování systémových prostředků samozřejmě ubírá z výkonu určitou režii a bylo by chybou ji bez pádného důvodu zvyšovat.

Pro sledování výkonu procesoru je podstatný čítač % času procesoru, který udává míru využití procesoru mezi dvěma posledními vzorky měření. Je zcela normální, když tato hodnota dosáhne 100 % v okamžiku spouštění aplikací nebo provádění náročných výpočtů, ovšem dlouhodobě by neměla přesáhnout 80 %. Pak je zřejmé, že právě procesor je slabým článkem systému. Druhým zajímavým ukazatelem může být čítač Přerušení/s. Dává přehled o tom, kolik hardwarových přerušení průměrně procesor za sekundu obslouží. Obvykle se jeho hodnota pohybuje okolo 1000, pokud je patrný nárůst, může to znamenat poruchu hardwaru a je třeba identifikovat, které zařízení přerušení generuje. Pro určení instance objektu slouží index procesoru, ten je u jedno-procesorových počítačů vždy 0. Při monitorování více-procesorových počítačů lze navíc využít instanci _Total, kdy budou čítače podávat souhrnné informace o všech procesorech.

Objekt operační paměti má dva důležité čítače. Čítač Bajty k dispozici (vyskytují se i odvozené čítače Počet kB k dispozici a Počet MB k dispozici) udává množství fyzické operační paměti, kterou může systém přidělit aplikacím. Doporučuje se mít alespoň 4 volné MB. Druhým čítačem je Stránky/s, je to počet stránek paměti čtených z disku nebo zapisovaných na disk. Objekt paměť je jednou z několika výjimek, u kterých nejsou žádné instance k dispozici – ani by to nemělo smysl.

U disku je třeba rozhodnout, zda-li chceme sledovat fyzický, nebo logický disk. Oba tyto objekty poskytují stejné čítače s tím rozdílem, že u logického disku lze navíc sledovat množství volného místa (čítače % volného místa a Volné megabajty). Důležitý ukazatel, který je společný pro fyzický i logický disk je čítač % času disku, který stejně jako u procesoru, znázorňuje průměrné vytížení disku mezi dvěma naposled měřenými vzorky. Podobně čítač Střední délka fronty disku udává průměrný počet diskových operací, které čekají ve frontě. Jako mezní hodnota pro čítač % času disku se udává 90 % a diskové operace by měly být ve frontě maximálně dvě. Pokud zjistíme, že je disk příliš vytížený, ještě to neznamená, že je slabým místem systému. Je třeba zkontrolovat volnou paměť a stránkování – právě to bývá často

příčinou velkého vytížení disku a na vině je příliš malé množství volné paměti. Pro rozlišení instance slouží u logických disků jejich název, např. `C:`. U fyzických disků je to index stejně jako u procesoru. Opět lze použít instanci `_Total` jak pro logické, tak pro fyzické disky.

Poslední součástí systému, kterou je třeba sledovat pro získání ucelených informací o jeho výkonu, jsou síťové adaptéry. Objekt `Rozhraní sítě` poskytuje např. čítače `Přijaté bajty/s`, `Odeslané bajty/s`, `Přijaté pakety/s` a `Odeslané pakety/s`. Názvy těchto čítačů jsou samopopisné a také neexistují žádné doporučené hodnoty – záleží na rychlosti síťového adaptéru. Jako instance se uvádí celé jméno síťového adaptéru (nikoli připojení). Tentokrát ale chybí instance `_Total`.

1.2 Nástroje pro monitorování systémových prostředků

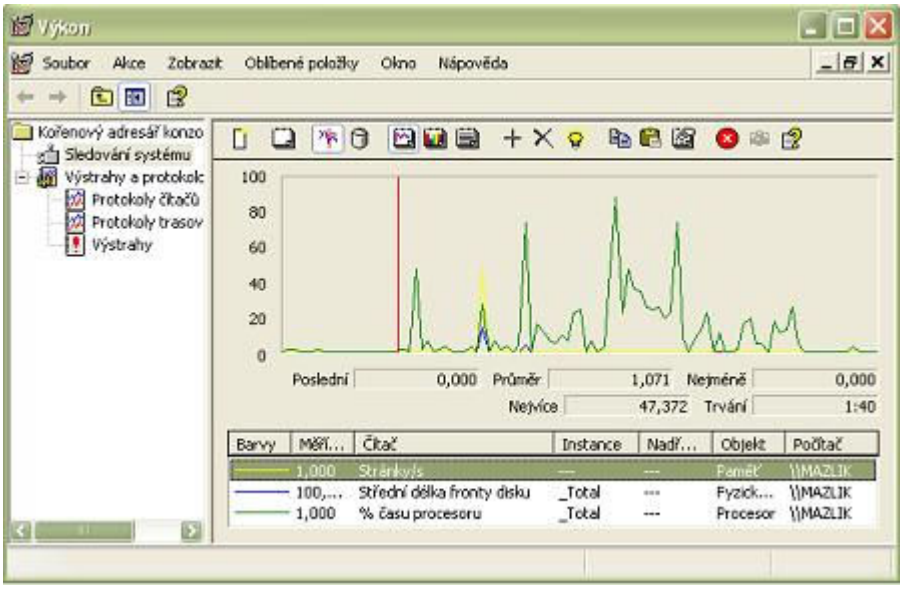
Od Windows 2000 dále jsou uživatelům a správcům k dispozici dva nástroje umožňující monitorování systémových prostředků. Kromě nich lze samozřejmě použít libovolné další nástroje, pro sledování běžících procesů je vynikající např. Process Explorer.

1.2.1 Správce úloh

Správce úloh je spíše uživatelský nástroj umožňující sledování aplikací a procesů aktuálně spuštěných v počítači. Poskytuje o procesech řadu dalších informací, jako využití procesoru, paměti nebo název uživatelského účtu, pod kterým je proces spuštěn. Mimo to podává statistiky o využití sítě, procesoru a paměti.

1.2.2 Konzola Výkon

Konzola Výkon je jednou z mnoha volitelných součástí MMC (Microsoft Management Console) konzoly. Tento nástroj, využívaný spíše administrátory než běžnými uživateli, se skládá ze dvou částí: Sledování systému a Výstrahy a protokolování výkonu. Sledování systému slouží k zobrazení informací o aktuálním výkonu nebo k prohlížení souboru protokolu. Umožňuje data zobrazovat ve formě grafu, histogramu nebo textové zprávy. Konzola poskytuje také statistické údaje o sledovaných objektech – u každého čítače udává minimální, maximální a průměrnou hodnotu, měřeno od spuštění sledování daného čítače.



Obrázek 1 - Sledování systému, zobrazení jako graf

Výhodou konzoly Výkon oproti Správci úloh je možnost přesně nakonfigurovat, jaká data chceme sbírat. Lze si vybrat libovolné objekty a jejich čítače, uložit takovou konfiguraci a použít ji na jiném počítači. Výběr objektu, instance, čítače a případně vzdáleného počítače odpovídá textu kapitoly 1.1.2.

Obrázek 2 - přidání čítače výkonu

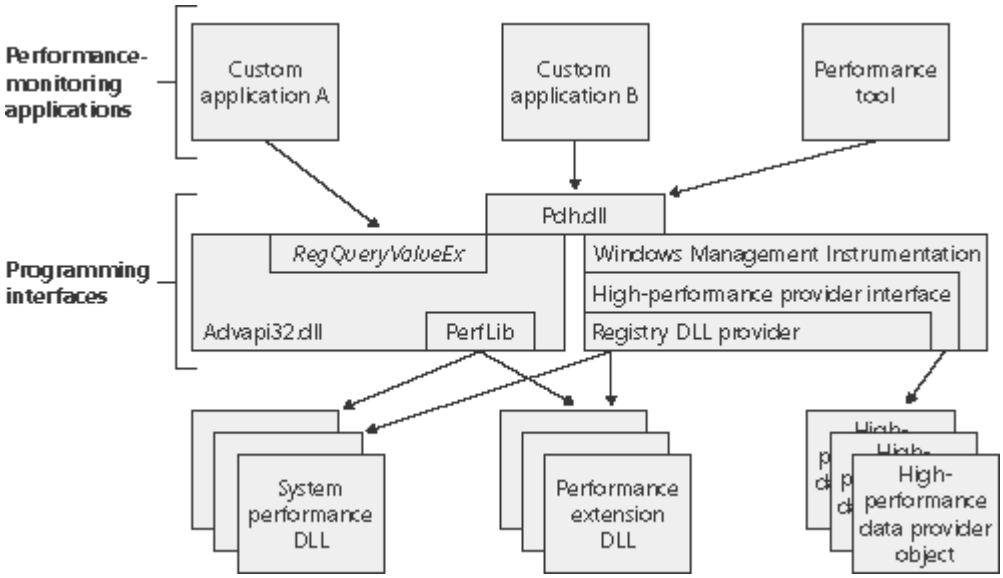
Výstrahy a protokolování slouží k vytvoření souboru protokolu. Opět lze nakonfigurovat libovolné objekty a čítače výkonu, zvolit jméno a typ souboru (textový, binární, SQL databáze...), interval sbírání dat a případně naplánovat, kdy se má záznam do souboru protokolu spustit a kdy ukončit.

2 Programové rozhraní sledování výkonu

V předchozí kapitole byly zmíněny dva stávající nástroje pro monitorování prostředků systému Windows. Pokud si chceme vytvořit vlastní nástroj, je situace poněkud komplikovanější. Existují celkem tři způsoby, jak získat data o výkonu.

První možností je přístup do registru. Čítače výkonu se nacházejí ve speciálním klíči HKEY_PERFORMANCE_DATA, který ovšem nelze nalézt v editoru registrů. Tento klíč je dostupný pouze programově, např. pomocí funkce `RegQueryValueEx`.. Přístup k čítačům přímo přes registry je pracnější a složitější než následující metody.

Další možností je použití knihovny PDH – Performance Data Helper, která rovněž přistupuje k čítačům přes registr, ale pro programátora je tento děj zcela transparetní. Kromě toho také zprostředkovává přístup k WMI (Windows Management Instrumentation) rozhraní. Zmíněné dva postupy ukazuje následující obrázek:



Obrázek 3 - Programové rozhraní sledování výkonu

Na obrázku je vidět aplikace A, která přistupuje k čítačům výkonu přes registry, dále Aplikace B a konzola Výkon, které využívají knihovnu PDH. Jako poslední možné řešení, které na obrázku chybí, je přímé využití zmíněného WMI rozhraní. Jak je vidět, toto rozhraní přistupuje ke stejným poskytovatelům dat o výkonu jako registry, navíc dokáže získávat informace z tzv. High Performance Data Providers – jedná se o čítače výkonu, které poskytují data mnohem rychleji. Takovéto

poskytovatele dat o výkonu lze dopsat do vlastní aplikace a rozšířit ji tak o možnost sledování jejích specifických parametrů.

2.1 Windows Management Instrumentation

Služba WMI slouží k usnadnění monitorování a správy celého systému. Vznikla jako implementace správy WBEM (Web – Based Enterpise Management). Programovací rozhraní navazuje na jádro služby, které spravuje jednotlivé poskytovatele dat. Tito poskytovatelé se vážou ke spravovanému objektu, což je hardwarové zařízení nebo aplikace reprezentovaná instancí WMI třídy. Přes programovací rozhraní navazuje ještě rozhraní skriptovací, využívané pro tvorbu administrativních skriptů a filtrů. Spolu se službou WMI je k dispozici řada již vytvořených tříd a poskytovatelů dat. Více informací lze nalézt v [4].

2.2 Knihovna Performance Data Helper

Tato knihovna vznikla jako nahrazení obtížnějšího přístupu k datům čítačů výkonu přes funkce registrů. Neslouží jen k získání informací ze stávajících čítačů, ale lze s její pomocí vytvořit i vlastní čítače výkonu. Protože tuto metodu jsem si vybral k vypracování aplikace, budu se jí věnovat podrobněji.

2.2.1 Vytvoření dotazu na čítače výkonu

Prvním krokem, který je třeba učinit pro získání údajů o výkonu touto metodou, je vytvoření dotazu. Dotaz je kolekce čítačů výkonu, přičemž v této kolekci se jednotně aktualizují data pomocí funkcí PDH knihovny. Dotazů může existovat více najednou, například lze mít dotaz obsahující čítače, které se budou aktualizovat každou vteřinu a jiný dotaz s čítači, u kterých postačí aktualizace několikrát za minutu. Dotaz dokáže nejen získávat aktuální údaje systému, ale je možné ho použít i pro čtení souboru protokolu. Pro vytvoření dotazu se používá funkce `PdhOpenQuery`. Bližší informace o této i následujících funkcích jsou k nalezení v [5].

2.2.2 Přidání čítačů do dotazu

Pokud máme vytvořený dotaz, můžeme do něj přidávat jednotlivé čítače výkonu. To se děje pomocí funkce `PdhAddCounter`. Této funkci se musí předat řetězec adresující konkrétní čítač, v případě sledování lokálního počítače je to `\objekt(instance)\čítač`. To, co se zdá být snadným úkolem, může programátora překvapit. Například při zadání řetězce `\Processor(0)\ % Processor Time` dojde k chybě a podle návratového kódu funkce `PdhAddCounter` lze zjistit, že požadovaný čítač neexistuje. Na čítače výkonu se totiž vztahuje lokalizace prostředí. Názvy

všech existujících čítačů se spolu s jejich indexy nachází v registru Windows HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\LangId. Zde je klíč s anglickými názvy a klíč s názvy dle lokálního prostředí uloženými ve více-řetězcové hodnotě Counter. V případě anglické verze Windows je přítomen jen klíč s anglickými názvy. Tento problém lze obejít použitím funkcí `PdhEmunObjects` a `PdhEmunObjectItem`, které poskytují názvy všech objektů a čítačů dle lokálního prostředí. To je ale použitelné jen v tom případě, že chceme sledovat úplně všechny objekty a jejich čítače. Zpravidla ale požadujeme sledování jen některých objektů a čítačů a tehdy je potřeba řešit lokalizaci přístupem do registru a zjištěním správných názvů.

Situaci usnadňuje funkce `PdhLookupPerfNameByIndex`, která na základě zadaného indexu čítače zjistí jeho název v lokálním jazykovém prostředí. Pokud bychom chtěli přistupovat k čítačům výkonu různých aplikací, stejně se práci s registrem nevyhneme. Indexy čítačů totiž nemusí být v každém systému stejné, nemění se pouze u čítačů hardwarových zařízení. V takovém případě je třeba načíst hodnotu Counter z klíče pro anglickou lokalizaci, vyhledat podle známých anglických názvů odpovídající indexy a ty potom předat funkci `PdhLookupPerfNameByIndex` ke zjištění platného názvu dle lokálního jazykového prostředí. Pro sestavení celé adresy čítače lze potom využít funkce pro práci s řetězci. Existují ještě další postupy sestavování adresy čítače, více se o nich dozvíte v [5].

2.2.3 Získávání hodnot

Po přídání čítačů do dotazu už lze získávat jejich hodnoty. Napřed ale musíme dát čítačům pokyn, aby se aktualizovaly, což se děje voláním fukce `PdhCollectQueryData`. Aktualizace je provedena globálně pro všechny čítače obsažené v daném dotazu. Zde ještě poznámka k instancím objektů. Pokud byl při přidávání čítače zadán špatný název instance, funkce `PdhAddCounter` na to nijak neupozorní, stačí syntakticky správný zápis adresy čítače, platné jméno objektu a čítače a přidání čítače považuje za úspěšné. Zareaguje teprve funkce `PdhCollectQueryData` chybovým kódem `PDH_NO_DATA`, a to ještě jen v případě, že v dotazu neexistuje žádný platný čítač. Jestliže dotaz alespoň jeden regulérní čítač obsahuje, funkce proběhne úspěšně.

Jakmile jsou hodnoty aktualizovány, lze přistoupit k jejich získání. To provádí např. funkce `PdhGetFormattedCounterValue`. Právě tato funkce odhalí výše popsanou situaci, kdy nebyla správně zadána instance objektu, protože provádí získání hodnoty konkrétního čítače. Odpovídajícím návratovým kódem pro chybějící instanci objektu je `PDH_INVALID_DATA`. Výhodou zmíněné funkce je, že při jejím volání můžeme specifikovat formát hodnoty čítače. Na výběr je formát double, long integer a 64-bitový integer.

Vytvoření dotazu a přidání čítačů výkonu je jednorázová akce, zatímco aktualizace a sbírání hodnot se děje periodicky. Po skončení práce s čítači výkonu by se měla zavolat funkce

`PdhRemoveCounter` pro každý sledovaný čítač a nakonec uzavřít dotaz pomocí funkce `PdhCloseQuery`.

2.2.4 Záznam do logovacího souboru

Záznam do logovacího souboru patří ke sledování výkonu a knihovna PDH k tomu poskytuje vlastní přístup. Nejprve je potřeba vytvořit dotaz na čítače výkonu, jak je popsáno výše. Následuje volání funkce `PdhOpenLog` pro otevření nebo vytvoření logovacího souboru (lze upřesnit parametry) a potom periodické volání funkce `PdhUpdateLog`. Rovněž lze zvolit typ souboru – CSV (hodnoty oddělené čárkami), TSV (hodnoty oddělené tabulátory), binární soubor a SQL databáze. Po skončení záznamu je třeba zavřít soubor i dotaz na čítače výkonu.

2.3 Další funkce Windows API

Přestože čítače výkonu poskytují značné množství informací o využití systémových prostředků, v některých případech je nutné nebo vhodné použít jiné řešení.

2.3.1 Procesy

Kromě sledování základních hardwarových prostředků (procesor, paměť, disky a síťová rozhraní) je výhodné mít rovněž přehled o aktuálně spuštěných procesech. K procházení běžících procesů slouží dvě knihovny – Tool Help Library a Process Status API.

Tool Help Library je modernější přístup, s podporou 64-bitových Windows. Jedná se o vytvoření „snímku“ současného stavu procesů, reprezentovaného seznamem struktur `PROCESSENTRY32`. V této struktuře nalezneme informace jako PID procesu, PID rodičovského procesu, jméno spouštěcího souboru a počet vláken. Knihovna poskytuje funkce pro vytvoření a procházení tohoto seznamu, navíc umožňuje i procházení jednotlivých modulů a vláken.

Knihovna Process Status API je zastaralejší, ale stále používaná. Tomu odpovídá i skutečnost, že oproti Tool Help Library nepodporuje 64-bitová Windows, ale na druhou stranu funguje i pod Windows NT. Bohužel získání stejných informací o procesu je o něco komplikovanější. Místo seznamu struktur obdržíme voláním funkce `EnumProcesses` pole PID. Tuto funkci je potřeba volat dvakrát, poprvé zjistí požadovanou velikost pole, poté provedeme alokaci a zavoláme funkci podruhé. Takovýto přístup předávání kolekce informací je ve Windows API poměrně běžný. Teprve teď začíná sbírání informací, které za nás udělá Tool Help Library. Pomocí PID každého procesu je třeba získat jeho handle, sloužící jako vstupní parametr funkcí pro získání jména spouštěcího souboru procesu, počtu vláken atd.

3 Analýza a návrh aplikace

Tato kapitola se věnuje analýze požadavků na vlastní nástroj pro monitorování systémových prostředků, návrhu této aplikace a její struktuře.

3.1 Analýza požadavků

Stejně jako v případě nástrojů uvedených v kapitole 1.2, se požaduje schopnost monitorovat výkon základních součástí systému – procesoru, paměti, logických nebo fyzických disků a síťových adaptérů. Rovněž je požadováno zobrazení informací o běžících procesech. Je třeba vybrat některé podstatné čítače výkonu a doplnit je dalšími užitečnými údaji. Neopomenutelnou součástí řešení je i přehledné zobrazení těchto informací uživateli. Navíc je u aplikace tohoto typu vhodné logování do souboru alespoň základních údajů.

3.2 Návrh aplikace

Před začátkem práce na vývoji aplikace je třeba si rozmyslet, jaký programovací jazyk a vývojové prostředí bude nejvhodnější. Výběr použitých knihoven a také programovacího paradigmatu je stejně důležitý.

3.2.1 Použité nástroje

Jako implementační jazyk jsem zvolil jazyk C++. Jazyk C je sice rovněž použitelný, ale přibyl by problém neustálého předávání mnoha parametrů mezi funkcemi, což se snadno vyřeší použitím atributů tříd. V těchto jazycích je také nejsnadnější přístup k funkcím Windows API. Použití jazyků Java nebo C# jsem zavrhl z důvodu nižší úspornosti paměti a relativně pomalému běhu aplikací pod platformami Java Runtime Environment a .NET Framework.

Práce bez použití vhodného integrovaného vývojového prostředí je dnes nemyslitelná. Vzhledem k vybranému implementačnímu jazyku připadaly v úvahu z obecně známých prostředí C++ Builder firmy Borland, volně dostupné Dev - C++ a nakonec Microsoft Visual Studio, které bylo pro tuto práci vybráno. Oproti ostatním vyniká v nápovědě při psaní kódu, v návrhu zdrojů a především v debuggeru, který umožňuje během krokování například nahlížet na hodnoty členů strukturovaných typů.

Spolu s jazykem C++ bylo použito Windows API bez jakýchkoli objektových nástaveb typu MFC. Tak jako tak, použití čistého API se v tomto projektu prakticky nelze vyhnout a jednotný ráz aplikace přispívá z hlediska programátora k její větší přehlednosti.

3.2.2 Členění aplikace

Kromě důvodu uvedeného na začátku předchozí kapitoly, návrh objektové struktury aplikace vnáší do projektu více přehlednosti. Celá aplikace se skládá z osmi tříd. Třída `Main` je vstupním bodem programu, obsahuje metody pro obsluhu uživatelského rozhraní a odkazy na objekt třídy `Tray`, `Setting` a `Log`. Třída `Tray` slouží k zobrazení tray ikony aplikace, třída `Setting` v sobě nese nejrůznější nastavení aplikace a sdružuje objekty dalších tříd – `Disc`, `Network`, `ProcMem` a `Process`. Stará se také o načtení a zápis některých z těchto nastavení do konfiguračního souboru.

Třída `Log` zajišťuje záznam informací o výkonu systému do logovacího souboru. Jelikož i při logování byla použita knihovna PDH, nabízí se několik formátů souboru – viz kapitola 2.2.4. Jako nejlepší možnost byl zvolen formát CSV. Jednak je srozumitelný už na první pohled při otevření v běžném textovém editoru, navíc je podporován např. aplikací MS Excel, kde je uživateli zobrazen ve formě tabulky. Nabízela by se možnost sestavit z této tabulky libovolný graf a dát tak zjištěným údajům co nejlepší přehlednost, ale bohužel způsob uložení hodnot v souboru, stejně jako jeho interpretace aplikací MS Excel, vytvoření grafu dosti komplikuje. Tento problém bude podrobně rozebrán v kapitole 5.5.

Třída `Disc` zjišťuje údaje o logických discích, konkrétně najde disky, které jsou v systému přítomné, určí velikost a volné místo vybraného disku, střední délku fronty a aktuální vytížení disku. Totéž je obsaženo ve třídě `Network` s tím rozdílem, že zjišťuje aktivní síťová rozhraní a počty odeslaných a přijatých bajtů za sekundu. Třída `ProcMem` monitoruje vytížení procesoru a paměti. Třída `Process` detekuje aktuálně běžící procesy a zjišťuje u každého z nich využití procesoru, paměti, PID a název procesu. Všechny tyto třídy se starají o aktualizaci a výpis sledovaných hodnot.

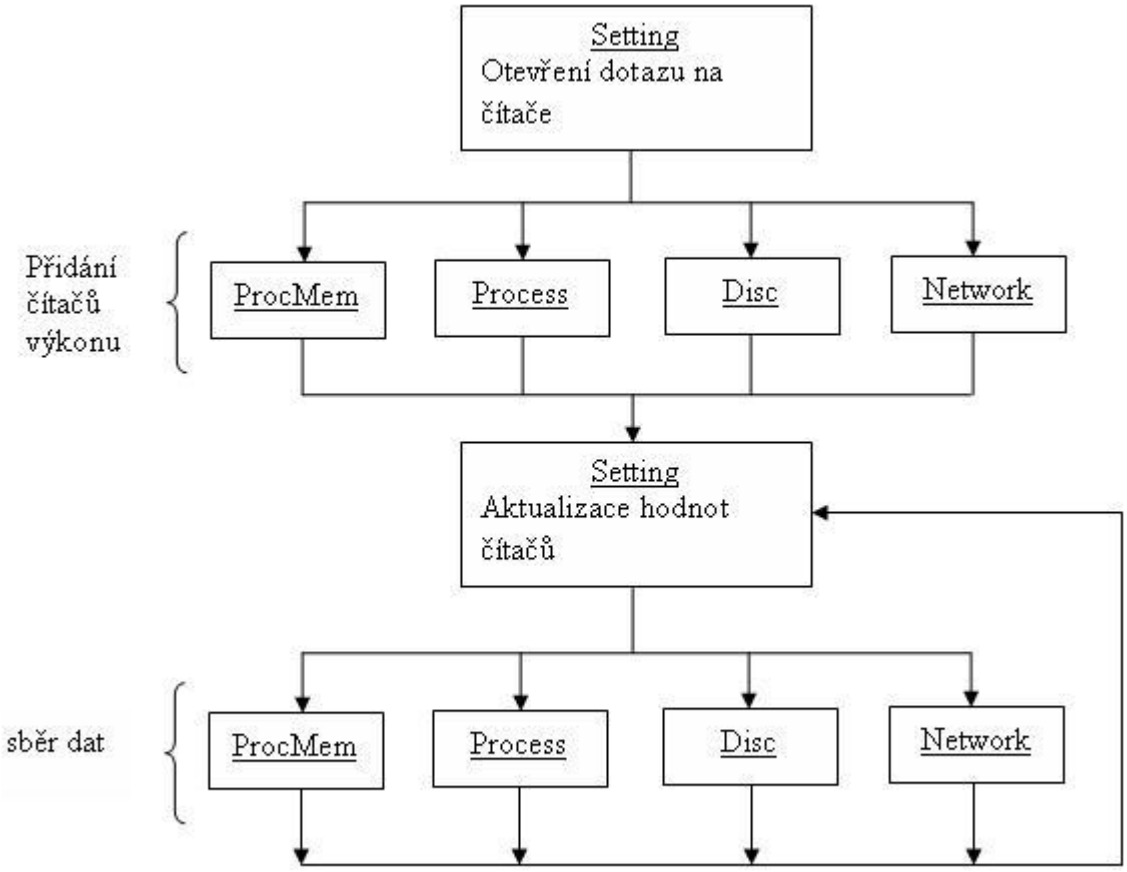
3.2.3 Konfigurační soubor

Vetšina aplikací si potřebuje některé údaje o nastavení uchovávat v konfiguračním souboru. V tomto případě tomu není jinak. Některé údaje – jako pozice přehledového okna na ploše – se uchovávají pro zvýšení uživatelské vstřícnosti aplikace, jiné prostě proto, aby aplikace fungovala. Mezi takové údaje patří například povolení logování, jméno logovacího souboru nebo disk a síťový adaptér, který si uživatel vybral ke sledování při minulém spuštění aplikace.

Konfigurační soubor má název setting.ini a je umístěn ve stejné složce, jako spouštěcí soubor aplikace. V případě jeho absence jsou zvoleny přednastavené údaje a nový konfigurační soubor je vytvořen při ukončení aplikace. Formát INI souboru je plně postačující, protože obsahuje jen několik řádků. V případě většího počtu položek by bylo vhodné použít XML.

3.2.4 Získávání údajů

Na obrázku 4 je vidět diagram získávání dat o výkonu. Ve Třídě `Setting` se vytvoří dotaz na čítače výkonu, tyto čítače jsou spravovány jednotlivými třídami odpovídajícími sledovaným zařízením, v nichž dochází k přidání požadovaných čítačů výkonu do dotazu.



Obrázek 4 - diagram získávání dat o výkonu

Následuje nekonečný cyklus složený ze dvou kroků – aktualizace hodnot čítačů a jejich čtení. Aktualizace se děje ve třídě `Setting` – probíhá pro všechny čítače patřící k dotazu, zatímco čtení hodnot se provádí jednotlivě v třídách `Disc`, `Process` atd. Na stejném principu funguje i logování do souboru, jen je potřeba vytvořit další dotaz na čítače z důvodu rozdílné periody aktualizace údajů.

3.2.5 Návrh GUI

Návrh uživatelského rozhraní bývá odsouván na vedlejší kolej, ale přehledný výpis i zadávání údajů je mnohdy pro uživatele cennější než rychlost a paměťová nenáročnost programu. Proto byla vytvořena dvě okna – přehledové, s průhledným pozadím, kde se uživateli zobrazují základní údaje, a hlavní okno aplikace. Důležitou roli hraje tray ikona, přes jejíž menu lze zobrazit hlavní okno a také

povolit přesunutí přehledového okna. To je jinak pevně ukotveno na ploše, aby uživatele nerušilo při práci.

Hlavní okno se skládá ze záložek, přičemž každá odpovídá sledovanému zařízení (resp. systémovému zdroji). Procesy jsou vypsány do seznamu o několika sloupcích. Toto řešení sice není nijak originální, ale v přehlednosti patrně patří mezi nejlepší. Hlavní okno navíc obsahuje menu pro některé další volby - zapínání logování, změna periody aktualizace údajů atd.

4 Popis implementace

Tato kapitola blíže rozvádí principy popsané v předchozí kapitole, upozorňuje na některé funkce Windows API, které jsou pro řešení významné, na problémy, které při vývoji nástroje vznikly, a na jejich řešení.

4.1 Použité knihovny

Při vývoji každého složitějšího programu se nelze obejít bez linkování knihoven. Mnoho z nich přilinkuje vývojové prostředí automaticky, aniž o tom víme. Jiné knihovny musíme přidat na vstup linkovacího programu sami.

Common Controls (comctl32.lib) obsahuje některé pokročilejší prvky grafického uživatelského rozhraní, například komponenty `List Control` (seznam o několika sloupcích) a `Tab Control` (záložky), které byly při vývoji aplikace použity.

Performance Data Helper (pdh.lib) je knihovna sloužící jako programovací rozhraní čítačů výkonu.

IP Helper (Iphlpapi.lib) ukrývá funkce a datové typy pro zjišťování informací o síťových rozhraních, pro zobrazení a změnu jejich konfigurace.

Process Status API (PSAPI) poskytuje rozhraní pro práci s procesy.

4.2 Spuštění a inicializace aplikace

Vstupní bod aplikace – funkce `WinMain` se nachází v souboru `Main.cpp`. Po spuštění je vytvořena instance třídy `Main` a obě okna. Nejsou to však klasická okna vytvořená registrací třídy okna a následně voláním funkce `CreateWindow`, ale dialogy. Pro vytváření dialogů se totiž používá funkce `CreateDialog`, která na rozdíl od `CreateWindow` dokáže použít navržený dialog ze souboru zdrojů. Přehledové okno je vytvořeno jako průhledné a navíc ignoruje zasílané zprávy od myši. Tím je docíleno efektu, že uživatel má kdykoli základní informace o výkonu systému k dispozici, ale neruší ho při práci zbytečně otevřené okno navíc. Okno může být totiž umístěno přes ikony na ploše, které jsou stále dostupné, jako kdyby žádné okno neexistovalo.

Následuje vytvoření instance třídy `Tray` – a tím i zobrazení tray ikony aplikace. Hlavní okno je po spuštění skryto, zobrazit se dá právě pomocí této tray ikony. Dále se vytváří instance třídy

`Setting` a volá se její inicializační metoda, ve které vznikají instance tříd pro sledování jednotlivých zařízení. V této metodě se také načítá konfigurační soubor, spouští se časovač a vytváří dotaz na čítače výkonu. Volají se i inicializační metody a funkce nezbytné pro správné fungování některých prvků uživatelského rozhraní. Pokud je v konfiguračním souboru povoleno logování dat o výkonu do souboru, je vytvořena také instance třídy `Log`.

Odkazy na objekty tříd `Log`, `Tray` a `Setting` jsou uloženy jako atributy v objektu třídy `Main`, jehož odkaz je veden jako globální proměnná – z důvodu dostupnosti pro metody obsluhy zpráv. Funkce pro obsluhu zpráv má totiž pevně stanovené parametry a pokud ji zkusíme deklarovat jako členskou nestatickou metodu nějaké třídy, pokusí se jí překladač při jejím volání předat dalším parametrem ukazatel `this`, což skončí chybou. Taková metoda tedy musí být deklarována jako statická. Potom samozřejmě nemá přístup k ukazateli `this` a je nutno hledat jinou možnost, jak z ní přistupovat k požadovaným datům. Například použitím globální proměnné.

4.2.1 Konfigurační soubor

Soubor setting.ini obsahuje tyto údaje: perioda aktualizace údajů, pozice x a y přehledového okna, index vybraného disku, index vybraného síťového adaptéru, povolení logování, perioda záznamu do logovacího souboru, jméno tohoto souboru a jeho index.

Během načítání souboru se přepíšou přednastavené hodnoty hodnotami ze souboru, provede se přesun přehledového okna. Při ukončení aplikace se tyto hodnoty zapíší zpět do souboru.

4.2.2 Procesor a operační paměť

Do konstruktoru třídy `ProcMem` se předává odkaz na dotaz na čítače výkonu, který je uložen ve třídě `Setting`, a handle přehledového okna – ten je zapouzdřen ve třídě `Main`. Handle dialogu na záložce v hlavním okně se do objektu třídy `ProcMem` předává až v okamžiku vzniku dialogu.

Nyní dochází k přidání čítačů výkonu. Pro procesor se přidává čítač `% času procesoru`, pro paměť je to `Počet MB k dispozici` a `Stránky/s`. Nejprve je potřeba najít správné názvy objektů a čítačů dle lokálního jazykového prostředí, k čemuž je použita funkce `PdhLookupPerfNameByIndex`. Indexy objektů a čítačů jsou v programu pevně nastaveny, získání jména z indexu bylo vysvětleno v kapitole 2.2.2. Postup je takový, že vytvoříme výsledný řetězec se zpětným lomítkem na začátku a pomocí funkce `PdhLookupPerfNameByIndex` zjistíme objekt `Procesor` – index 238. Připojíme jméno objektu do výsledného řetězce a přidáme do kulatých závorek instanci procesoru, v tomto případě `(_Total)`. Uvedení této instance místo čísla procesoru zaručí správné fungování i na více-procesorových počítačích. Informace budou podávány souhrnně, kdybychom chtěli rozlišit využití jednotlivých procesorů, bylo by třeba přidat čítač pro každou instanci. Dále přidáme znak zpětného lomítka a znovu použijeme funkci

PdhLookupPerfNameByIndex, které tentokrát předáme index 6 - % času procesoru. Takto jsme vytvořili řetězec \Procesor(_Total)\% času procesoru, který bude mít například na anglické lokalizaci Windows podobu \Processor(_Total)\% processor time. Nyní stačí zavolat funkci PdhAddCounter, které se předá dotaz na čítače výkonu, ukazatel na handle přidávaného čítače a řetězec adresující čítač a program je schopen monitorovat využití procesoru.

Stejným postupem se vytvoří i čítače pro operační paměť. Index objektu paměť je 4, index čítače Počet MB k dispozici je 1382, čítače Stránky/*s* 40. Řetězce předávané funkci PdhAddCounter jsou potom \Paměť\Počet MB k dispozici a \Paměť\Stránky/s. U objektu Paměť se žádná instance zařízení neuvádí.

Handle čítačů výkonu jsou uloženy jako atributy třídy reprezentující dané zařízení. Jsou totiž nutné v momentě, kdy chceme z čítače přečíst hodnotu. Při sestrojování řetězce adresujícího čítač výkonu je nezbytné použít řetězec dvou-bajtových znaků a potom i odpovídající modifikace funkcí pro práci s řetězci. Windows API poskytuje datové typy WCHAR a TCHAR, které jsou definovány následujícím způsobem:

```
#ifdef UNICODE
 typedef WCHAR TCHAR;
#else
 typedef char TCHAR;
#endif

typedef wchar_t WCHAR;
```

Typ wchar_t reprezentuje dvou-bajtový znak, typ WCHAR je z něho přímo odvozen, zatímco typ TCHAR je v případě, že je definována symbolická konstanta UNICODE, typem WCHAR, jinak je definován jako jedno-bajtový typ char. Funkce pro práci s dvou-bajtovými znaky se v použití nijak neliší od těch tradičních pro znaky jedno-bajtové, jejich popis je k nalezení v [12].

4.2.3 Logické disky

Sledování logických disků je obsaženo ve třídě Disc. V jejím konstruktoru se jednak předává odkaz na dotaz na čítače výkonu a handle přehledového okna, ale především je provedeno zjištění připojených logických disků v systému pomocí funkce GetLogicalDriveStrings. Tato funkce ve výstupním parametru předá řetězec obsahující jména disků jako C:\, D:\ atd. Jména disků jsou od sebe oddělena nulou ('\0'). Údaj v takovémto formátu by byl pro další zpracování dosti nepraktický, jednotlivé disky jsou proto umístěny do pole čtyřznakových řetězců.

Drobným problémem je, že funkce `GetLogicalDriveStrings` naplňuje předávaný řetězec všemi logickými jednotkami, proto je potřeba vybrat jen jednotky logických disků. Funkce `GetDriveType` dokáže zjistit, zda se jedná o pevný disk, CD-ROM mechaniku, odnímatelný disk apod.

Pro zjištění velikosti a volného místa na disku je použita funkce `GetDiskFreeSpaceEx`. Tato funkce používá pro předání velikostí datový typ `ULARGE_INTEGER`, což je strukturovaný typ obsahující ve dvou částech nižší a vyšší čtyři bajty osmi-bajtového celého čísla a to samé osmi-bajtové číslo vcelku. Je to z toho důvodu, že ne každý kompilátor dokáže přeložit tak dlouhé celé číslo. Tímto ovšem nezjistíme volné místo na celém logickém disku, ale volné místo disku pro uživatele, který spustil monitorovací nástroj. Funkce `GetDiskFreeSpaceEx` totiž respektuje přidělené kvóty. To platí jen pro souborový systém NTFS, systémy FAT32 a FAT16 kvóty nemají.

Jelikož se do přehledového okna vypisuje jen volné místo na vybraném logickém disku, přičemž tato hodnota se získává funkcí popsanou v předcházejícím odstavci, nejsou zatím čítače výkonu potřeba. Ty budou přidány až v okamžiku vytvoření dialogu na odpovídající záložce hlavního okna aplikace.

4.2.4 Síťové adaptéry

Monitorování síťových adaptérů je obsaženo ve třídě Network. Stejně jako u předchozích dvou tříd je i zde v konstruktoru předán odkaz na dotaz na čítače výkonu a handle přehledového okna. Podobně jako v případě logických disků, je potřeba zjistit, jaké síťové adaptéry se v systému nacházejí. Zde se objevil problém.

Názvy nainstalovaných síťových adaptérů se nacházejí v registru Windows v klíči HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkCards. Zde každému adaptéru náleží podklíč, ve kterém je uvedeno jméno a popis adaptéru. Jméno adaptéru je jeho jednoznačný identifikátor, například {40DB726E-2681-4A41-9949-CD23C89E19C7}, zatímco popis řekne člověku víc: Realtek RTL8139 Family PCI Fast Ethernet NIC. Jakmile přidáme nějaké čítače výkonu pro objekt `Rozhraní sítě`, zjistíme při pokusu o aktualizaci hodnot čítačů, že byla zadána nesprávná instance zařízení (bylo naznačeno v kapitole 2.2.3). Ani identifikátor, ani popis nelze použít. Pokud otevřeme konzolu Výkon, zjistíme, že požadované jméno instance je v tomto případě Realtek RTL8139 Family PCI Fast Ethernet NIC _2 - Packet Scheduler Miniport. Zde _2 je číslo nainstalovaného síťového adaptéru a Packet Scheduler Miniport přidává do názvu ovladač síťové karty.

Windows API poskytuje funkci `GetAdaptersInfo` pro zjištění informací o konfiguraci adaptérů. Tuto funkci je třeba volat dvakrát podobně jako funkci `EnumProcesses`, o níž byla zmínka v kapitole 2.3.1, a která bude ještě popsána v následující kapitole. Funkce `GetAdaptersInfo` navrací ve výstupním parametru seznam struktur `IP_ADAPTER_INFO`,

kde nalezneme celou řadu informací, například fyzickou adresu, IP adresu, adresu výchozí brány, DHCP server a podobně. V seznamu jsou zahrnuty pouze povolená síťová rozhraní. Zahrnuje to i připojení přes VPN, ale jen v případě, že je spojení v daném okamžiku aktivní. Pro nás je důležitý údaj Description – tedy popis stejně jako v registrech. Zde už získáme kompletní název, téměř shodný s hledaným názvem: Realtek RTL8139 Family PCI Fast Ethernet NIC #2 - Packet Scheduler Miniport. Jediným rozdílem je znak '#' oproti '_'. Během testování na jiných počítačích bylo odhaleno více podobných rozdílů – například znak '/' opět nahrazen znakem '_' nebo kulaté závorky nahrazeny hranatými. Bohužel seriózní řešení nebylo v rámci Windows API nalezeno a po přednesení tohoto problému na diskuzních skupinách MSDN byl odpovědí jen odkaz na WMI rozhraní. Problém je tedy řešen záměnou některých speciálních znaků v názvu adaptéru.

Po zjištění správného názvu adaptéru už není problém přidat čítače výkonu, konkrétně `\Rozhraní sítě(adaptér)\Přijaté bajty/s` a `\Rozhraní sítě(adaptér)\Odeslané bajty/s`.

4.2.5 Procesy

Konstruktor třídy Process obsahuje oproti jiným třídám jednu důležitou akci – úpravu oprávnění procesu monitorovací aplikace. Je to z toho důvodu, že pokud chceme získávat informace o systémových procesech, musíme mít alespoň oprávnění pro ladění. Každý proces má svůj Access Token – seznam aktuálně platných oprávnění, který je odvozen od uživatelského nebo systémového účtu, pod kterým je proces spuštěn. Nejprve se získá handle Access Tokenu procesu, což řeší funkce `OpenProcessToken`. Dál je třeba získat LUID – 64-bitový identifikátor oprávnění, kterým je v tomto případě `SeDebugPrivilege`. Tento identifikátor spolu s informací, zda chceme oprávnění povolit či zakázat, vložíme do struktury `TOKEN_PRIVILEGES`, kterou předáme spolu s handle Access Tokenu funkci `AdjustTokenPrivileges`. Ta provede změnu požadovaného oprávnění.

Dále obsahuje konstruktor volání metody `SetCounters`, která je narozdíl od stejnojmenných metod ostatních tříd výrazně složitější. Informace jako jméno procesu, PID a spotřeba paměti lze dostat voláním různých funkcí API, ale pro určení využití procesoru je potřeba použít čítače výkonu. Musíme tedy sestavit řetězec `\Proces(jméno)\% času procesoru`, kde `jméno` je název spouštěcího souboru procesu bez přípony.

Pro každý proces jsou uchovávány výše zmíněné údaje a navíc handle čítače výkonu pro využití procesoru. Alternativa, kdy by se uchovávalo minimum informací je nepřípustná, protože při každém požadavku na aktualizaci dat (v krajním případě každou vteřinu) by se muselo provádět odebrání a znovu přidání čítače pro každý proces, což by aplikaci zpomalovalo.

Ke zjištění aktuálně běžících procesů je použita funkce `EnumProcesses`, která navrací ve výstupním parametru ukazatel na pole identifikátorů procesu (PID). Z dalšího zpracování je

úmyslně vynechán proces s PID 0, který je ve správci úloh zobrazen jako „nečinné procesy“. Pomocí funkce `OpenProcess` se získá handle procesu a funkcí `EnumProcessModules` dostaneme moduly procesu, což jsou jeho spustitelné soubory nebo dynamicky linkované knihovny. Voláním funkce `GetModuleBaseName` obdržíme jméno hlavního modulu – tedy např. explorer.exe. Získané jméno uložíme do pole informací o procesech a po odstranění přípony použijeme k adresaci čítače využití procesoru.

4.2.6 Logování

Při vytvoření instance třídy `Log` dochází k otevření dalšího dotazu na čítače výkonu. Do něj jsou přidány čítače `% času disku` pro všechny logické disky, `Bajty celkem/s` pro všechna síťová rozhraní, `Počet MB k dispozici` operační paměti a `% času procesoru` objektu `Procesor`. Takto připravený dotaz na čítače výkonu se potom předává do funkce `PdhOpenLog` spolu s názvem logovacího souboru, typem souboru, omezením jeho maximální velikosti a dalšími volbami, které určují, jak se bude do souboru zapisovat.

Na výběr je otevření souboru pro čtení (`PDH_LOG_READ_ACCESS`), zápis (`PDH_LOG_WRITE_ACCESS`) a aktualizaci (`PDH_LOG_UPDATE_ACCESS`), další volba specifikuje, zda se má vždy vytvořit nový soubor (`PDH_LOG_CREATE_NEW`), použít existující (`PDH_LOG_OPEN_EXISTING`) nebo otevřít cyklický soubor (při dosažení maximální velikosti souboru se bude od začátku přepisovat – tato volba je dostupná pouze při použití binárního souboru). Kromě těchto základních voleb otevření souboru existují ještě jejich kombinace, více lze nalézt v [7].

Původní myšlenka byla zapisovat neustále do jednoho souboru – vytvořit nový, pokud neexistuje, jinak přidávat záznamy na konec existujícího souboru. Nabízí se tedy využít volby `PDH_LOG_CREATE_NEW` v kombinaci s `PDH_LOG_WRITE_ACCESS` pro nový soubor a `PDH_LOG_OPEN_EXISTING` spolu s `PDH_LOG_UPDATE_ACCESS` pro zápis na konec existujícího souboru. Zatímco zápis do nového souboru fungoval dle očekávání, přidání dalších záznamů se již nepodařilo. Funkce `PdhOpenLog` vrátila chybový kód označující, že tato funkce není implementována. Pro ověření byl nakonfigurován protokol čítačů v konzole Výkon a třikrát za sebou spuštěn a vypnut. Byly vytvořeny tři soubory, k jejichž názvu přibylo pořadové číslo. Z toho se dá usuzovat, že firma Microsoft v knihovně PDH zřejmě vůbec neřeší posun na konec logovacího souboru při jeho dalším otevření. Protože práce se souborem je při použití této knihovny skryta, není možnost si pomoci například funkcí `fseek`. Použít pouze první zmíněný přístup k otevření souboru také nelze – soubor je při druhém otevření přepisován od začátku, takže bylo zvolené stejné řešení jako u firmy Microsoft – v konfiguračním souboru se kromě jména uchovává i index logovacího souboru, který je při každém otevření ke jménu připojen a zvýšen o jedna.

4.3 Běh aplikace, sledování výkonu

V předchozí kapitole bylo popsáno, co se děje v aplikaci při jejím spuštění – inicializace, nastavení monitorování. Nyní se zaměříme na běh programu a jeho chování při interakci s uživatelem.

4.3.1 Aktualizace údajů o výkonu

Pro třídy `Network`, `Disc` a `ProcMem` je postup získání dat o výkonu téměř stejný. Ve třídě `Main` dochází v reakci na událost časovače k volání metody `TimerProc` třídy `Setting`. Tato metoda nejprve volá funkci `PdhCollectQueryData`, jejímž jediným argumentem je handle dotazu na čítače uložený rovněž ve třídě `Setting`. Tím dochází k aktualizaci hodnot všech čítačů výkonu, které byly do dotazu přidány při jeho vytváření.

Dále jsou volány metody `Update` jednotlivých tříd – `Proc`, `MemDisc`, `Network`, ale i `Process`, u které je postup komplikovanější a bude lépe rozebrán v kapitole 4.3.2. Každá z metod `Update` obsahuje volání funkce `PdhGetFormattedCounterValue`, která poskytuje hodnotu konkrétního čítače pro zvolený formát (formátem je datový typ, ve kterém hodnotu požadujeme, většinou `LONG` nebo `double`). Pokud se některé hodnoty získávají jinak než použitím čítačů výkonu, jsou rovněž volány v metodě `Update`. Tak je tomu u třídy `Disc` – získání velikosti a volného místa pomocí `GetDiskFreeSpaceEx` a u třídy `ProcMem` – zjištění velikosti fyzické a virtuální paměti funkcí `GlobalMemoryStatus`. Všechny získané hodnoty jsou uloženy do odpovídajících atributů jednotlivých tříd.

Nyní již stačí údaje zobrazit. O to se starají metody `Print` zmíněných tříd, které jsou volány hned po metodách `Update`.

Aktualizace logovacího souboru je řešena stejně, avšak provádí se separátně, protože logování je řízeno jiným časovačem. V reakci na událost tohoto časovače je volána metoda `Update` třídy `Log`, v níž je obsaženo pouze volání funkce `PdhUpdateLog`, která se postará o zápis záznamu do souboru.

4.3.2 Aktualizace Procesů

Metoda `Update` třídy `Process` nejprve zkontroluje, zda se od jejího minulého volání změnil počet procesů. Stačí jednou zavolat funkci `EnumProcesses` a podle toho, kolik bajtů paměti požaduje alokovat pro pole PID, lze určit aktuální počet procesů. Pokud se počet procesů od minula změnil, volají se metody `DeleteCounters` (odebere se všem procesům v seznamu čítač využití procesoru z dotazu na čítače výkonu), `SetCounters` (znovu se projdou všechny procesy a přidají se čítače) a `Print` (zobrazí se všechny procesy). Jestliže se počet procesů nezměnil, projdou se všechny procesy

a pro každý se volá funkce `GetProcessMemoryInfo`, která podává souhrn informací o paměti procesu. Z těchto informací je vybrána `WorkingSetSize`, což je množství paměti aktuálně přidělené procesu. Kromě toho se pro každý proces přečte hodnota jeho čítače využití procesoru (handle čítače je součástí informací o procesu, které uchováváme).

Nyní přichází na řadu další optimalizace – do komponenty `List Control` jsou vypsány jen ty procesy, jejichž využití procesoru nebo paměti se od poslední aktualizace změnilo. Ostatní procesy zůstávají v komponentě `List Control` beze změny. Toto opatření je vynuceno použitím zmíněné komponenty a bude zdůvodněno v následující kapitole.

4.3.3 Uživatelské rozhraní

Bez grafického uživatelského rozhraní si lze takovouto aplikaci jen těžko představit a některé jeho části vyžadují bližší popis. Většina metod spadajících do interakce s uživatelem je obsažena ve třídách `Main` a `Tray`. V ostatních třídách je pouze metoda `Print`, která zapisuje na dialogy údaje o výkonu.

Po spuštění aplikace je uživateli zobrazeno pouze přehledové okno a tray ikona. Přes menu tray ikony lze aplikaci ukončit, povolit přemístění přehledového okna (pomocí zapnutí a vypnutí ignorování zpráv od myši) a zobrazit hlavní okno aplikace.

V hlavním okně se kromě menu nachází komponenta `Tab Control`, která vytváří záložky odpovídající sledovaným zařízením. Komponenta se ovládá zasíláním zpráv, pokud tedy chceme vytvořit čtyři záložky, pošleme jí čtyři zprávy, v nichž bude parametrem adresa struktury, která obsahuje informace o vytvářené záložce. Na každé záložce je umístěn dialog, ten se vytváří z šablony dialogu umístěné do paměti při inicializaci této komponenty. Pro čtyři záložky je tedy třeba pole čtyř šablon dialogů. Kdykoli potom uživatel zvolí některou ze záložek, je zrušen dialog na minulé a vytvořen nový dialog na aktuální záložce. Handle tohoto nově vytvořeného dialogu je nutno předat do třídy, která odpovídá za zařízení, jehož data o výkonu jsou na záložce zobrazována, aby metoda `Print` této třídy byla schopna na dialog zobrazovat údaje. Každý dialog má svoji vlastní metodu pro obsluhu zpráv.

Další zajímavou komponentou je `List Control` sloužící k zobrazení běžících procesů. Stejně jako `Tab Control` se ovládá zasíláním zpráv. Při vytvoření dialogu záložky o procesech se tato komponenta inicializuje zasláním zpráv o přidání sloupců a voláním metody `Print` třídy `Process`, která vypíše do komponenty informace o všech procesech opět zasíláním zpráv. Zde přichází na řadu zdůvodnění optimalizací při výpisu procesů. Naprosto nepřípustné je při každé aktualizaci všechny řádky z komponenty odstranit a znovu je vypsat. Komponenta doslova bliká a aplikace reaguje tak pomalu, že je téměř k nepoužití. Proto se aktualizuje pouze řádek o procesu, jehož využití paměti nebo procesoru se změnilo.

Na dialozích záložek `Síť` a `Disky` je umístěna komponenta `ComboBox`, do níž se při inicializaci daného dialogu vypíše seznam síťových adaptérů, resp. logických disků. Pokud uživatel vybere některý disk nebo adaptér, dojde k odebrání čítačů výkonu minulého disku či adaptéru a k přidání čítačů vybraného.

Logování do souboru se zapíná a vypíná na dalším dialogu, který je dostupný přes menu hlavního okna. Zapnutí či vypnutí logování znamená vytvoření nebo zrušení instance třídy `Log` a změnu frekvence aktualizace souboru a povolení logování v instanci třídy `Setting`. Jestliže se jedná jen o změnu frekvence aktualizace, je pouze přenastaven časovač logování.

4.4 Ukončení aplikace

Při ukončení aplikace je zaznamenána poloha přehledového okna a spolu s dalšími nastaveními zapsána do konfiguračního souboru. Následuje volání destruktorů vytvořených objektů, ve kterých se provádí odebrání čítačů výkonu, případně zrušení časovačů nebo celého dotazu na čítače výkonu.

5 Testování

Testování je součástí životního cyklu vývoje softwaru. Co přesně se musí otestovat, je různé u každé aplikace. Dobrým pomocníkem pro testování jsou stávající nástroje – například zmíněný Správce úloh a konzola Výkon, kdy se pouze porovnají odpovídající údaje s naší aplikací. Stačí tedy vyvolat situace, které povedou ke změně údajů o výkonu.

5.1 Procesy

Ověřuje se, zda program správně rozpoznal všechny běžící procesy a dokáže zobrazit i procesy systémové. Dále je třeba vyzkoušet chování při spuštění či ukončení nějakého procesu – jestli je do seznamu přidán nebo z něj odebrán. Posledním krokem je zatížení některého procesu a zjištění, zda jeho využití procesoru a paměti vzrostlo (pro přesnost je nutné porovnat s jiným nástrojem).

5.2 Procesor a paměť

Vytížení procesoru se kontroluje snadno – stačí zatížit libovolný proces a sledovat, jak se ukazatel změnil. Platí, že celkové vytížení je dáno součtem využití procesoru jednotlivými procesy. Údaje o paměti bych doporučil ověřit porovnáním s programem Správce úloh, případně spuštěním nějaké aplikace a zjištěním, jestli se dostupná paměť zmenšila o stejné množství, jaké bylo přiděleno nově spuštěné aplikaci. Údaj Počet stránek/s lze otestovat spuštěním paměťově náročných aplikací – při nedostatku volné paměti se zvýší odkládání pamětových stránek na disk.

5.3 Síťové adaptéry

Množství dat přenesených po síti se nejlépe zkontroluje přidáním odpovídajících čítaču výkonu v konzole Výkon a porovnáním získaných hodnot.

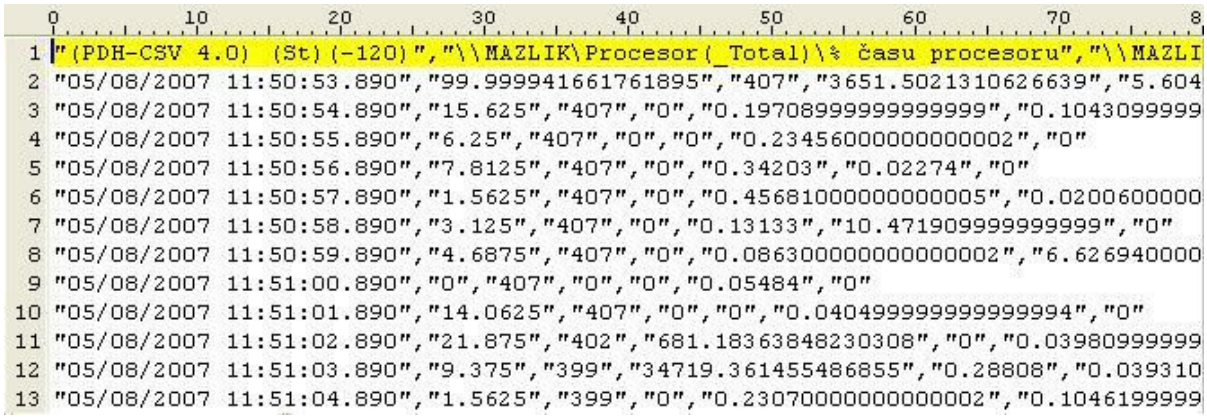
5.4 Logické disky

Velikost a množství volného místa každého logického disku lze snadno zjistit ve vlastnostech každého disku v nabídce Tento počítač. Využití disku a fronta jeho úloh se dá otestovat kopírováním většího množství dat. Využití disku se bude pohybovat kolem 100 % a ve frontě bude jedna úloha (pokud provedeme více kopírování dříve, než první skončí, bude fronta úloh delší).

5.5 Logování

Zde je potřeba zkontrolovat, jestli funguje logování po spuštění aplikace (pokud je povoleno v konfiguračním souboru), dále jestli ho lze za běhu vypnout a znovu spustit. Při každém zapnutí logování by měl být vytvořen nový soubor, za jehož název je připojen index. Rovněž je třeba ověřit, zda funguje změna frekvence aktualizace souboru. Samozřejmostí je ověření obsahu souboru.

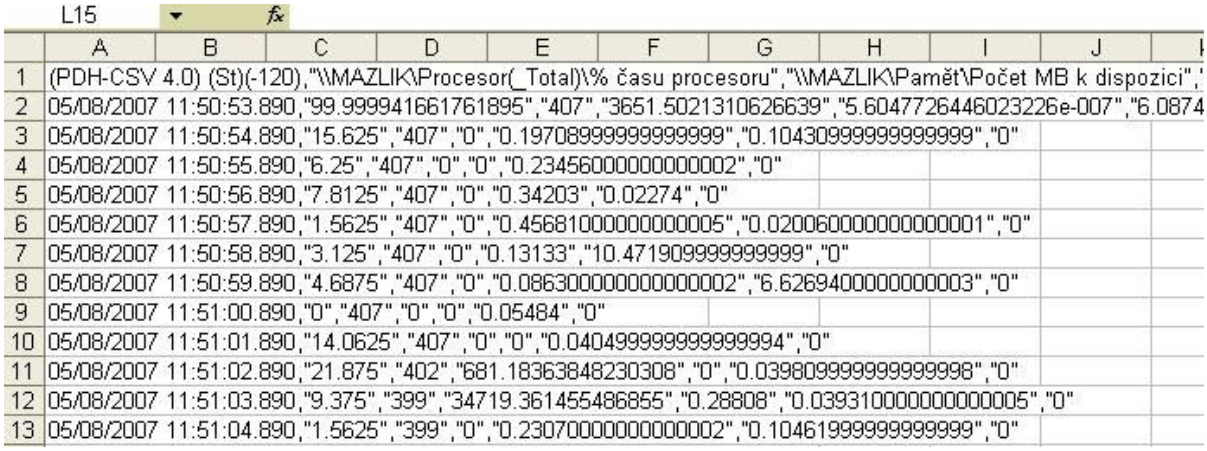
Na obrázku 5 vidíme obsah logovacího souboru. První řádek informuje, jaké čítače výkonu byly zaznamenány. Na dalších řádcích je datum a čas pořízení záznamu a hodnoty čítačů uvedené ve stejném pořadí, v jakém jsou čítače vyjmenovány v prvním řádku.



```
1  "(PDH-CSV 4.0)  (St)(-120)","\\MAZLIK\Procesor(_Total)\% času procesoru","\\MAZLI
2  "05/08/2007 11:50:53.890","99.999941661761895","407","3651.5021310626639","5.604
3  "05/08/2007 11:50:54.890","15.625","407","0","0.19708999999999999","0.1043099999
4  "05/08/2007 11:50:55.890","6.25","407","0","0","0.23456000000000002","0"
5  "05/08/2007 11:50:56.890","7.8125","407","0","0.34203","0.02274","0"
6  "05/08/2007 11:50:57.890","1.5625","407","0","0.45681000000000005","0.0200600000
7  "05/08/2007 11:50:58.890","3.125","407","0","0.13133","10.471909999999999","0"
8  "05/08/2007 11:50:59.890","4.6875","407","0","0.086300000000000002","6.626940000
9  "05/08/2007 11:51:00.890","0","407","0","0","0.05484","0"
10 "05/08/2007 11:51:01.890","14.0625","407","0","0","0.040499999999999994","0"
11 "05/08/2007 11:51:02.890","21.875","402","681.18363848230308","0","0.03980999999
12 "05/08/2007 11:51:03.890","9.375","399","34719.361455486855","0.28808","0.039310
13 "05/08/2007 11:51:04.890","1.5625","399","0","0.23070000000000002","0.1046199999
```

Obrázek 5 - náhled logovacího souboru v textovém editoru

Obrázek 6 ukazuje ten samý soubor otevřený v aplikaci MS Excel 2003. Nevýhoda je patrná na první pohled. Knihovna PDH zapisuje hodnoty do souboru v uvozovkách a MS Excel navíc tento soubor načte tak nešťastně, že celý řádek umístí do jediné buňky. Nebýt těchto dvou problémů, dal by se snadno ze zaznamenaných dat sestrojit graf. Takto se vytvoření grafu značně komplikuje.



L15

	A
1	(PDH-CSV 4.0) (St)(-120),"\\MAZLIK\Procesor(_Total)\% času procesoru","\\MAZLIK\Paměť\Počet MB k dispozici","'
2	05/08/2007 11:50:53.890,"99.999941661761895","407","3651.5021310626639","5.6047726446023226e-007","6.0874
3	05/08/2007 11:50:54.890,"15.625","407","0","0.19708999999999999","0.10430999999999999","0"
4	05/08/2007 11:50:55.890,"6.25","407","0","0","0.23456000000000002","0"
5	05/08/2007 11:50:56.890,"7.8125","407","0","0.34203","0.02274","0"
6	05/08/2007 11:50:57.890,"1.5625","407","0","0.45681000000000005","0.020060000000000001","0"
7	05/08/2007 11:50:58.890,"3.125","407","0","0.13133","10.471909999999999","0"
8	05/08/2007 11:50:59.890,"4.6875","407","0","0.086300000000000002","6.6269400000000003","0"
9	05/08/2007 11:51:00.890,"0","407","0","0","0.05484","0"
10	05/08/2007 11:51:01.890,"14.0625","407","0","0","0.040499999999999994","0"
11	05/08/2007 11:51:02.890,"21.875","402","681.18363848230308","0","0.039809999999999998","0"
12	05/08/2007 11:51:03.890,"9.375","399","34719.361455486855","0.28808","0.039310000000000005","0"
13	05/08/2007 11:51:04.890,"1.5625","399","0","0.23070000000000002","0.10461999999999999","0"

Obrázek 6 - náhled logovacího souboru v MS Excel 2003

6 Závěr

Cílem práce bylo vytvoření uživatelsky příjemného nástroje, který mě bude neustále informovat o výkonu počítače, aniž by nějak rušil při práci. Poskytuje většinu stejných informací, jako Správce úloh a navíc některé další, které jsou obecně při sledování výkonu systému považovány za podstatné. Aplikace byla rozšířena o možnost logování údajů do souboru, což je u nástroje pro monitorování systémových prostředků více než vhodné. Druhým cílem bylo poznat operační systém Windows z pohledu programátora a naučit se používat jeho API.

Při tvorbě tohoto nástroje byl dodržen postup v souladu se zásady softwarového inženýrství. Nechybí analýza, návrh ani testování aplikace.

Již při dokončování práce bylo jasné, že řadu věcí šlo udělat jiným, vhodnějším způsobem a řadu dalších věcí by bylo dobré do programu ještě přidat. Mezi podstatnější změny a návrhy na vylepšení se dá zahrnout konfigurovatelnost aplikace – dát uživateli možnost, aby si sám vybral nějaký čítač výkonu nebo přizpůsobil uživatelské rozhraní (například změna barev, jazyková lokalizace ovládacích prvků apod.). Tím by vznikla další řada údajů, které by se musely uložit do konfiguračního souboru, pro který by byl tentokrát zvolen formát XML. Dále se nabízí možnost přepracovat sledování procesů – využít knihovnu Tool Help Library a rozšířit poskytované informace o procesech. Rovněž by bylo vhodné zohlednit nástup procesorů s více jádry a monitorovat každé jádro zvlášť. Vzhledem ke špatně řešenému logování knihovnou Performance Data Helper je třeba zvážit realizaci nového systému logování dat.

Literatura

[1] Microsoft Corporation Microsoft Windows XP Professional MCSA/MCSE Training Kit. Praha, Computer Press 2002.

[2] Microsoft Corporation Microsoft Windows 2000 Server Správa systému. Praha, Computer Press 2000.

[3] Petzold, C.: Programování ve Windows. Praha, Computer Press 1999.

[4] WMI Architecture. MSDN Library. Dokument dostupný na URL: http://msdn2.microsoft.com/en-us/library/aa394553.aspx (5.5.2007).

[5] Creating a Query. MSDN Library. Dokument dostupný na URL: http://msdn2.microsoft.com/en-us/library/aa372013.aspx (5.5.2007).

[6] Using PDH APIs correctly in a localized language. Microsoft Knowledge Base. Dokument dostupný na URL: http://support.microsoft.com/kb/q287159/ (5.5.2007).

[7] Performance Counters Reference. MSDN Library. Dokument dostupný na URL: http://msdn2.microsoft.com/en-us/library/aa373088.aspx (5.5.2007).

[8] API function call unification: the case of Processes/Modules enumeration. The Code Project. Dokument dostupný na URL: http://www.codeproject.com/threads/processapi.asp (5.5.2007).

[9] Using ListView control under Win32 API. The Code Project. Dokument dostupný na URL: http://www.codeproject.com/listctrl/listview.asp (5.5.2007).

[10] Učíme se WinAPI. Builder. Dokument dostupný na URL: http://www.builder.cz/art/cpp/winapi1.html (5.5.2007).

[11] Enabling and Disabling Privileges in C++. MSDN Library. Dokument dostupný na URL: http://msdn2.microsoft.com/en-us/library/aa446619.aspx (5.5.2007).

[12] Alphabetical Function Reference. MSDN Library. Dokument je dostupný na URL: http://msdn2.microsoft.com/en-us/library/634ca0c2(VS.80).aspx (5.5.2007)

Seznam příloh

Příloha 1.

Uživatelská příručka

Radovan Synek, FIT VUT v Brně, 2007

Systémové požadavky

Aplikace je určena pro operační systém Windows XP. Testována byla na Windows XP SP2 CS, Windows XP SP2 EN a Windows Server 2003 R2 EN.

Kompilace

Zdrojové soubory (dále hlavičkové soubory a popisy zdrojů) jsou umístěny na CD (příloha 2) ve složce Source. Spolu s nimi jsou zde také soubory o konfiguraci projektu. Aplikace byla vyvíjena v MS Visual Studiu 2005.

Spuštění

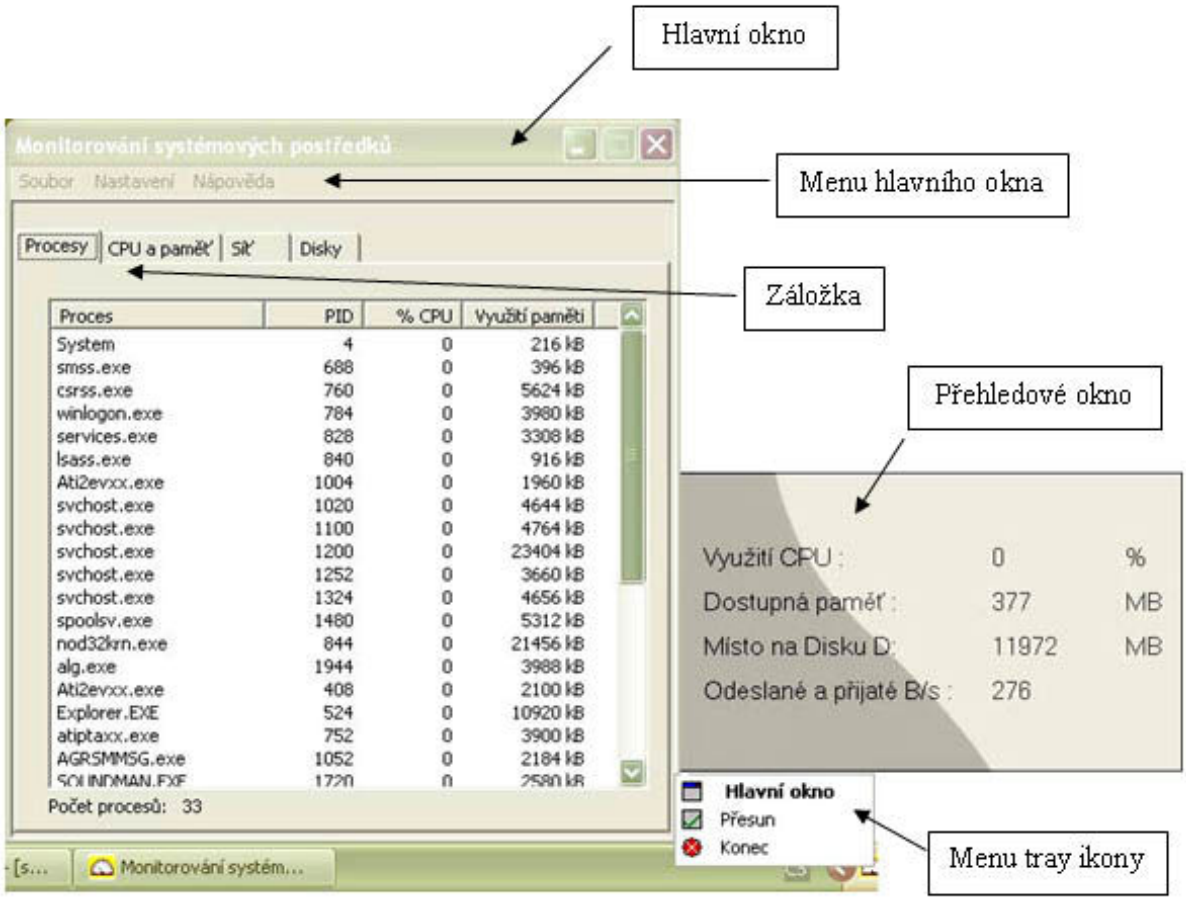
Spouštěcím souborem je BP_API.exe. Pokud chybí konfigurační soubor (musí být umístěn ve stejné složce jako spouštěcí soubor), zobrazí se varovná hláška. To ničemu nevadí a program bude fungovat. Při ukončení si sám vygeneruje nový konfigurační soubor.

Ovládání

Po spuštění je zobrazeno přehledové okno a tray ikona. Pokud na tray ikonu poklepete levým tlačítkem myši, zobrazí se hlavní okno aplikace. Jestliže na ni kliknete pravým tlačítkem, zobrazí se menu s těmito položkami:

- Hlavní okno — zobrazí hlavní okno aplikace
- Přesun — povolí nebo zakáže přesun přehledového okna
- Konec — ukončí aplikaci

Popis oken:



Menu hlavního okna aplikace:

- Soubor - Konec ukončení aplikace
- Nastavení - Frekvence nastavení frekvence aktualizace údajů
  - Záznam hodnot povolení logování do souboru
- Nápověda - O aplikaci zobrazí dialog s informacemi o aplikaci

Záložky:

Procesy

Zobrazuje přehled běžících procesů – jejich jméno, PID, využití procesoru a paměti.

CPU a paměť

Vypisuje a graficky znázorňuje využití CPU, dále informuje o velikosti fyzické a virtuální paměti a o jejich množství k dispozici. Tyto údaje jsou podávány v kB. Posledním údajem je počet paměťových stránek přenesených na disk za sekundu.

Síť

Podává informace o množství odeslaných a přijatých dat vybraného síťového rozhraní. Pro změnu sledovaného rozhraní otevřete rozbalovací nabídku, vyberte rozhraní a klikněte na tlačítko Vybrat.

Disky

Zobrazuje velikost, využité a volné místo zvoleného logického disku, jeho aktuální využití a frontu úloh. Pro změnu sledovaného disku otevřete rozbalovací nabídku, vyberte logický disk a klikněte na tlačítko Vybrat.

Dialog Aktualizace údajů:

Zobrazí se přes menu Nastavení – Frekvence. Šipkami lze změnit četnost aktualizace údajů v rozmezí jednou za 1 až 10 vteřin.

Dialog Logování:

Zobrazí se přes menu Nastavení – Záznam hodnot. Po povolení logování lze nastavit četnost aktualizace logovacího souboru v rozmezí jednou za 1 až 60 vteřin.

Konfigurační soubor:

Konfigurační soubor nese název setting.ini. Je umístěn ve stejné složce jako spouštěcí soubor.

Logování:

Jméno logovacího souboru je nastaveno v konfiguračním souboru setting.ini. Pokud tento údaj nebo celý soubor setting.ini chybí, je zvoleno jméno LogX.csv, kde X je pořadové číslo souboru.