Process Watcher

Děkujeme, že jste se rozhodli vyzkoušet náš program. Doufáme, že se Vám bude líbit. Co Process Watcher nabízí?

- Ü postartovní nastavení priorit všem běžícím procesům
- Ü získávání různých informací o aktivních procesech (spotřeba paměti, používané moduly, počet threadů, oken atd.)
- Ü ukazuje nejdůležitější statistiky o výkonu systému (zatížení procesoru, obsazení fyzické i virtuální paměti, velikost odkládacího souboru a jeho využití aj.)
- Ü umožňuje korekci priorit méně důležitých aplikací, které se o to nepostarají sami
- Ü dokáže sledovat spouštěné programy a přidělovat jim prioritu podle přání uživatele

To vše umožňuje Process Watcher. Výsledkem jeho snažení by měla být racionalizace využívání procesorového času aplikacemi, které se samy o sebe nedokáží postarat a nadužívají systémových zdrojů neúměrně svému vlastnímu významu, a tak brzdí z hlediska uživatele celý systém.

Pokud jste se rozhodli všechny tyto funkce vyzkoušet, věžte však, že Process Watcher je určen hlavně pokročilejším uživatelům, protože jeho nesprávným používáním můžete stabilitu systému narušit. Pokud se už k pokročilejším uživatelům počítáte, doufáme, že náš program oceníte. Tento manuál by Vám měl pomoci řešit všechny problémy a využít Watchera tak, jak si jeho autor představoval.

Konvence této nápovědy

Větší nebo významnější bloky (přibližně v rozsahu jedné kapitoly nebo tématu) jsou nadepsány modrým písmem. Menší souvislejší úseky textu jsou nadepsány červeně. Červená barva je použita i k zvýraznění tématu a upozornění na nový blok textu, který se věnuje jinému problému než blok předchozí. Bloky textu nadepsané červeně mohou být dále členěny, podtržené nadpisy jsou vyšší úrovně než pouze tučné písmo.
Popisky tlačítek a dalších ovládacích prvků jsou uváděny bez akcelerátoru (tj. s podtrženého znaku, který slouží jako zkratka při současném stisku *ALT*). Přesné názvy ovládacích prvků, oken apod., které se vyskytují v souvislém textu, jsou uváděny kurzívou.
Jména souborů jsou zásadně velkými písmeny. Klávesy jsou uvedeny velkými písmeny v kurzívě.

Rejstřík

V tomto rejstříku jsme připravili několik kapitol shrnujících několik základních okruhů, které by mohly naše uživatele zajímat, nebo které jim pomohou při práci s Watcherem.

Nároky na používání Watchera
Nejste si jisti, zda Watcher poběží na Vašem počítači? Chcete vědět, jaké nároky na systém si klade? Vaše otázky budou zodpovězeny v této kapitole.

Podmínky používání
Asi Vás to nebude zajímat, ale pro případ, že ano, pročtěte si za jakých podmínek můžete Watchera (legálně) používat. Už teď ale jedna dobrá zpráva: Vaše peněženka se nemusí bát – Watcher je freeware.

Instalace a odinstalace

Pokud Vás zajímají detaily instalace/odinstalace, či se vyskytly nějaké problémy, vše najdete právě zde.

Ovládání & přehled všech funkcí
Tak toto je asi to nejdůležitější – jistě bude chtít zjistit, k čemu slouží to či ono tlačítko.

How to…
…čili česky Jak na to… Nebaví Vás pročítat celé ovládání a potřebujete rychle najít právě jen tu jednu funkci?

Tipy (nejen pro dnešek)
Snad oceníte několik rad autora.

Známé problémy
Watcher stejně jako jiné programy trpí několika chybami (za některé třeba ani vůbec nemůže a jsou způsobeny Windows), nebo se někdy může chovat sice korektně, ale trošku jinak než si uživatel představuje. Než sáhnete do rubriky *Řešení problémů*, zkuste se podívat sem. Pokud objevíte nebo znáte řešení některého problému, dejte nám určitě vědět.

Řešení problémů
Když něco nepůjde podle Vašich představ, možná tu najdete řešení.

Slovníček pojmů a vysvětlivek
Pokud Vás v textu zarazí neznámé slovo nebo se chcete dozvědět o něčem něco navíc, zkuste se podívat do našeho malého slovníčku.

Pro odborníky a kolegy takyprogramátory
Pro odborníky a vůbec všechny, které zajímá, jak to vlastně funguje, si autor připravil malou exkurzi do zákulisí. Snad někoho poučíme, snad inspirujeme, ale především doufáme, že skutečný odborník, který sem zabloudí, nám poskytne nové nápady a připomínky k našemu řešení. K pročtení zájemcům doporučujeme i rubriku *Známé problémy*. Všechny nápady vítáme!

Ze zákulisí
Zajímá Vás něco o lidech, kteří toto mají na svědomí, něco z historie vývoje programu, nebo se chcete spojit s autorem, abyste ho obeznámili se svým nápadem, problémem nebo připomínkou?

Bonbónek na závěr
Obdivujeme vytrvalce, kteří dočetli až sem, a nedivíme se těm netrpělivým, kteří sem přeskočili po pár řádcích. Doufáme, že několik Murphyho zákonů o fungování počítačové techniky nevyzní jako alibistické zdůvodnění neplánovaných pochybení autora.

Na závěr ještě jednou rychle kontaktní adresy.

Nároky na používání Watchera

Naštěstí nároky na používání Watchera jsou malé: stačí jen fungující Windows 95 (nebo kompatibilní operační systém) a adaptér s rozlišením 800x600 bodů (doporučeno alespoň 256 barev) nebo větším. Na disku zaberou soubory do 2 MB.

Upozornění uživatelům jiných operačních systémů než Windows 95: na Vašich systémech nebyl Watcher testován a nezbývá nám (a Vám), než věřit firmě Microsoft, že tyto systémy jsou s Windows 95 kompatibilní. Na Vašich systémech především nelze úplně a slepě věřit informacím o spotřebě paměti procesy, např. Windows NT mají jiný mechanismus práce s pamětí a mapováním modulů do adresového prostoru procesů než Windows 95, a tak informace o záboru paměti moduly, které vybraný proces používá, budou značně podhodnocené.

Uživatelům Windows 98: Watcher byl testován na české verzi Windows 98, přičemž se přišlo na několik chyb, které byly v průběhu vývoje odstraněny a nepůsobí již potíže.

Detailnější rozbor nároků

Protože se Watcher snaží o zlepšení a zrychlení chodu systému, bylo věnováno velké úsilí, aby sám byl příkladem. Hlavní modul zabírá asi 72 KB. Modul uživatelského rozhraní, který zabírá okolo 512 KB, se nahrává do paměti pouze při aktivaci Watchera. Monitorovací modul, který se nahrává jen při zapnutém monitorování, spotřebuje 80 KB. Modul speciálních dialogů se používá jen výjimečně a tehdy zabere asi 596 KB.
Shrneme-li to: za normálního provozu spotřebuje Watcher zanedbatelných 72 KB, při zapnutém monitorování pak 152 KB. Připočtěme alokovanou paměť (kterou do velikosti modulů nepočítáme) a dostaneme se k maximální paměťové spotřebě v klidovém stavu do 170 KB, což není zas tak mnoho. Ovšem pozor, pokud chceme zapínat detailní informace nebo je dokonce získáváme na pozadí. V tomto případě nároky na paměť dočasně prudce rostou a mohou dosáhnout řádově až desítky megabyte (jsou to pouze odhady, avšak dosti reálné). V případě, že detailní informace zapneme pouze k aktuálnímu procesu, nemusí to ještě znamenat takovou zátěž, pokud ale získáváme detailní informace na pozadí a máme málo paměti, může snadno dojít k jejímu vypotřebování. Minimalizací Watchera bude všechna paměť opět uvolněna.

Zpomalení systému způsobené samotnou existencí procesu je při standardní nízké klidové prioritě Watchera zanedbatelné. Prioritu si Watcher zvyšuje jen při své aktivaci, aby odezva na uživatelovy požadavky byla přiměřeně rychlá. Určitou zátěž může představovat monitoring, zde je však těžké odhadnout jakou. Vzhledem k jeho koncepci, by tato zátěž také neměla být příliš veliká.

Pokud i by tyto nízké nároky (nebereme v úvahu detailní informace a jejich nároky) byly pro někoho příliš vysokou cenou za používání Watchera, lze jej používat v zástupcích jiných aplikací jako spouštěč nebo jej nechat po startu Windows nastavit priority vybraným procesům a pak jej nechat ukončit (viz funkce parametrů).

Podmínky používání

Předpokládá se, že každý uživatel Process Watchera je s těmito podmínkami srozuměn. Process Watcher je freeware s limitovanou podporou. To znamená, že jej uživatel smí (a dokonce by to bylo prospěšné) šířit program naprosto volně, **ale nesmí nijak zasahovat do obsahu žádného souboru, který celý programový balík tvoří** (netýká se souboru STATS.DAT, který může být libovolně měněn i smazán) **ani jej přejmenovávat, dále jej nesmí komerčně využívat bez souhlasu autora** (tj. např. prodávat – za komerční využití se nepovažuje zveřejnění na freewarovém či sharewarovém výběru nebo příloze časopisu; autor by ale byl rád o takovémto využití předem informován).

Autor neručí za žádné škody způsobené na počítači, na kterém je Process Watcher nainstalován.

Instalace a odinstalace

Instalace a odinstalace probíhá za pomoci instalačního programu (SETUP.EXE). Jediný problém, který by se mohl vyskytnout, lze předpokládat při odinstalaci ve Windows NT (viz poznámka níže).

Instalace

Instalace probíhá takto: zkopírujete všechny dodané soubory na disk do společné složky, kde se má Watcher instalovat. Spustíte program Setup. Nemáte-li novější verzi Process Watchera, můžete na úvodní dotaz odpovědět *Ano*, čímž vlastně instalaci vykonáte a otevře se Vám hlavní okno Setupu. V případě, že máte instalovanou novější verzi Process Watchera, řiďte se pokyny v manuálu této novější verze (pravděpodobně bude doporučena nebo přímo vyžadována odinstalace). Pokud používáte nějaký podobný program, není vyloučeno, že se budou oba programy navzájem rušit, a lze jen doporučit, abyste nepoužívali oba naráz.

V hlavním okně Setupu můžete nastavit několik voleb:

Pomoc v nesnázích...
V kapitole *Nároky na používání Watchera* byla už zmíněna skutečnost, že při testech na české verzi Windows 98 byla objevena závažná chyba lokalizátorů Windows. Následkem toho bylo nutno udělat mechanismus, který tuto chybu odstraní. V případě, že se Vám Watcher při pokusu o aktivaci ukončí s hláškou o fatální chybě v důsledku přejmenování statistik ve Vaší verzi Windows, můžete za pomoci Setupu této chybě čelit. Jednoduše klikněte na toto tlačítko a ze seznamu šablon v seznamu po levé straně dialogu vyberte svou verzi Windows a dialog uzavřete stiskem tlačítka *Ok*. Pokud se zde Vaše verze Windows nenachází, máte dvě možnosti: zkusíte jiné šablony, nebo vytvoříte vlastní šablonu editací souboru STATS.DAT, který je připraven k editaci v textovém poli v dolní polovině dialogu. V originálním souboru STATS.DAT je v podobě komentáře k dispozici stručný popis struktury souboru i návod, jak jej editovat. Nastavení šablony je samozřejmě globální, platí pro všechny uživatele.

Spouštět při startu Windows
Watcher se při startu Windows bude automaticky spouštět. Spouštěn bude vždy s parametrem /W, což zaručuje snadnou konfigurovatelnost pomocí dialogu *Konfigurace*. Narozdíl od předchozí verze se toto pole vztahuje nyní pouze k aktuálnímu uživateli.

Zástupce na pracovní ploše
Zaškrtnutím/odškrtnutím tohoto pole vytvoříte/zrušíte zástupce na pracovní ploše Windows. Tato volba je platná pouze pro aktuálního uživatele.

Skupina Process Watcher v nabídce Start
Podobně jako zástupce na pracovní ploše můžete vytvořit nebo zrušit skupinu *Process Watcher* v nabídce Start. Také tato volba ovlivní pouze aktuálního (přihlášeného) uživatele.

Odstranit konfigurační záznamy
Zruší konfigurační záznamy aktuálního uživatele – ztratí se tedy jeho nastavení Watchera. Zůstává mu ale skupina v menu Start a zástupce na pracovní ploše, pokud je vytvořil. Konfigurační záznamy se vytváří při otevření dialogu *Konfigurace*.

Kompletně odinstalovat
Zruší postupně zástupce, skupiny a konfigurační záznamy všech uživatelů Watchera na počítači, odstraní

tedy ze systému výsledky činnosti Watchera a Setupu. Pokud poté smažete složku se soubory Watchera, definitivně jej odstraníte ze svého počítače, jinak jej můžete opětovným spuštěním Setupu znova nainstalovat.

Poznámka pro uživatele Windows NT:
Pod Windows NT nebyl Watcher testován. Podle dostupných materiálů lze soudit, že mohou při odinstalaci vzniknout problémy s odstraňováním konfiguračních záznamů, je nutno ověřit si, zda byly opravdu odstraněny a popřípadě je odstranit ručně.

Poznámka pro předchozí verze:
Verze s číslem menším než 1.0 (týká se to verze 0.9, která byla omylem uveřejněna – viz historie vývoje) nejsou kompatibilní s verzí 1.01 a před její instalací by měly být odstraněny ze systému. U verze 1.0 toto není úplně potřeba: spusťte Setup verze 1.0 a zrušte položku *Spouštět při startu Windows* a pak vyměňte staré soubory za nové. V tomto případě zůstanou zachována uživatelská nastavení, s výjimkou startu při spuštění Windows, protože tato volba je platná nyní pouze pro jednoho uživatele a uživatelé by si měli zkontrolovat, zda mají vybránu možnost start Watchera při spuštění Windows (v Setupu) a jaké mají nastavení parametrem /W (dialog *Konfigurace*). Lze ovšem starou verz odinstalovat úplně a pak nainstalovat novou.

Dodané soubory

Instalace Process Watchera se skládá z několika souborů. V případě, že jste neobdrželi originální samorozbalovací archív, zkontrolujte, zda máte všechny potřebné soubory.

README.TXT	soubor se základními informacemi, které by uživatel měl znát před instalací
WATCHER.EXE	hlavní modul
WATCHER.DLL	modul s uživatelským rozhraním
SPECDLGS.DLL	modul se speciálními dialogy
MONHOOK.DLL	monitorovací modul
STATS.DAT	soubor s patchem pro některé verze Windows
SETUP.EXE	instalační program
WATCHER.HLP	soubor s nápovědou (manuál) – tento soubor
WATCHER.CNT	soubor s obsahem nápovědy

Odinstalace

Spusťte Setup (buďto přímo nebo z *Ovládacích panelů/Přidat nebo ubrat programy*). Pro zrušení záznamů jen přihlášeného uživatele stiskněte tlačítko *Odstranit konfigurační záznamy* a popřípadě zrušte zástupce a programovou skupinu. Pro kompletní odinstalaci stiskněte tlačítko *Kompletně odinstalovat*; soubory nebudou odstraněny z disku, jejich odstraněním odstraníte Watchera z počítače úplně, jinak jej můžete opět nainstalovat opětovným spuštěním Setupu.

Ovládání & přehled všech funkcí

Watcher používá několik dialogů. Zde je seznam těchto oken s podrobným popisem všech funkcí:

Hlavní okno
Konfigurace
Vlastnosti ukazatele
Nový proces

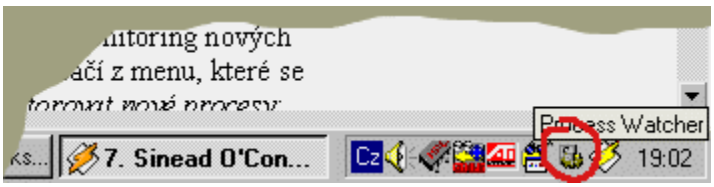
Při spouštění Watchera můžete použít i parametry.

Používání Watchera

Watchera lze používat několika způsoby, záleží na uživateli, aby si vybral ten, který mu nejvíce vyhovuje, popřípadě ten, který nejméně zatěžuje jeho počítač. Nezaregistrovaní uživatelé (tj. bez vlastních konfiguračních nastavení) nemohou některé funkce využít.

Watcher jako obyčejná aplikace na pozadí
Je to obvyklé (a také plánované) využití Watchera. Watchera jednoduše spustíte (nebo jej necháte spouštět automaticky při startu Windows) a dále se již o něj nemusíte starat. Jeho přítomnost je indikována ikonou v panelu úloh vedle hodin, zpravidla vpravo dole (tzv. tray ikona). Kliknutím pravým tlačítkem na tuto ikonu si zobrazíte menu s dostupnými funkcemi. Volba *Aktivovat* je ekvivalentní dvojkliku na ikonu a zobrazí Vám hlavní okno Watchera. Hlavní okno je možno vynést do popředí i jednoduchým kliknutím za současného stisku (a podržení) klávesy *SHIFT* nebo *CONTROL*, čímž vyvoláte hlavní okno v tzv. rychlém módu, kdy jsou nezávisle na uživatelském nastavení vypnuty některé funkce, které mohou příliš zatěžovat systém (jako např. automatické občerstvování seznamu procesů), což je užitečné, je-li počítač právě velmi zatížen a potřebujeme rychlejší odezvu. Používáme-li Watchera takto na pozadí, můžeme zapnout i monitoring nových procesů (máme-li monitoring aktivní a potřebujeme-li jej dočasně vypnout, stačí z menu, které se zobrazí při kliknutí pravým tlačítkem na tray ikonu, vybrat položku *Monitorovat nové procesy*; opětným výběrem této volby monitoring zase aktivujeme).

Tray ikony v panelu úloh vpravo dole na monitoru, tray ikona Watchera je zakroužkovaná:



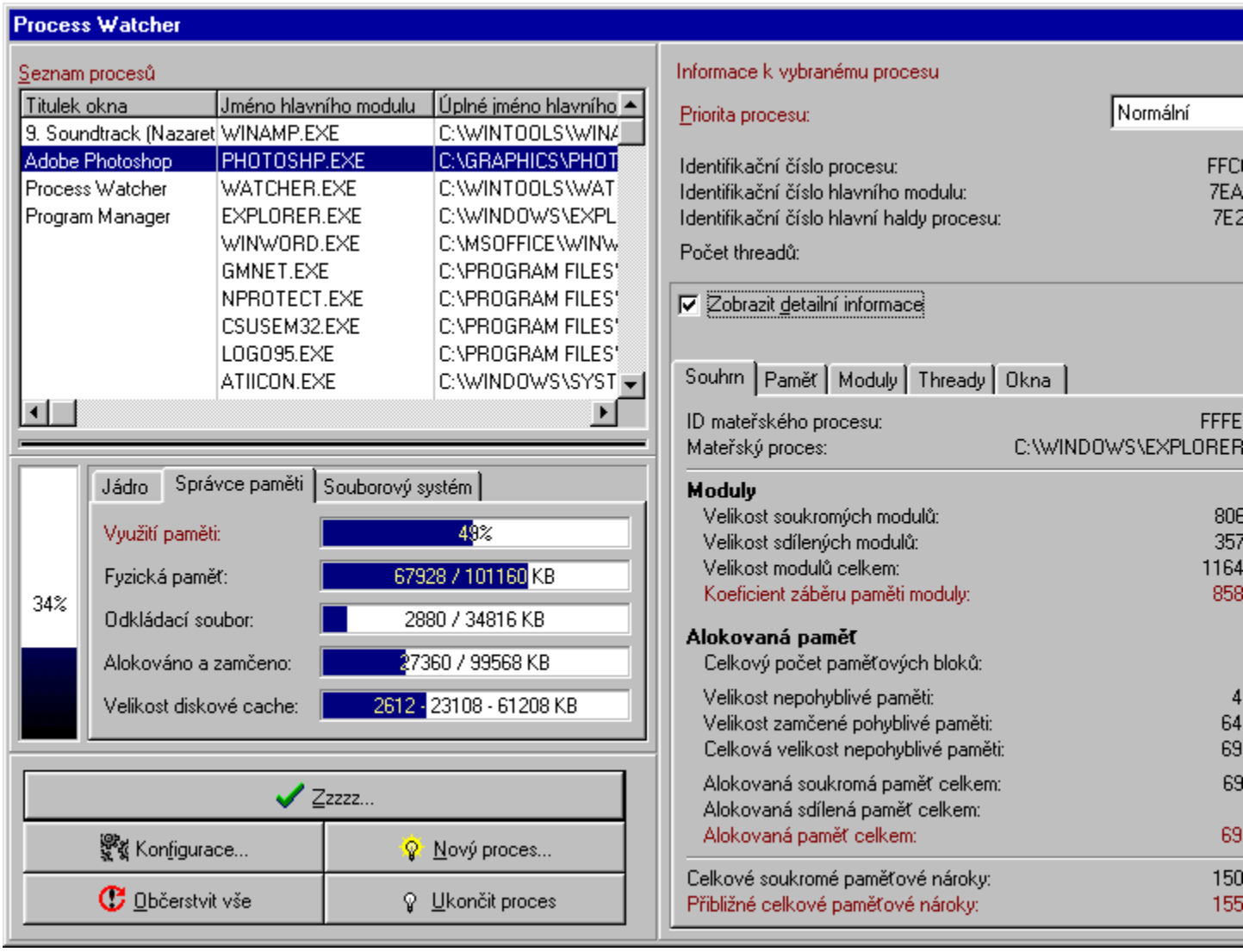
Nastavení priorit jiným aplikacím při startu Watchera
Takto mohou využívat Watchera jen registrovaní uživatelé. V dialogu *Konfigurace* si vytvoří seznam procesů kontrolovaných při startu Watchera (záložka *Monitorování procesů*), zapnou tuto funkci a nastaví postartovní prodlevu této kontroly (záložka *Obecná nastavení*). V případě, že chtějí využít tuto funkci i po startu Windows, musí zkontrolovat nastavení parametru /W (dolní rámeček na záložce *Obecná nastavení*), kterým mohou ovlivnit, zda má Watcher po startu s tímto parametrem pokračovat jako aplikace v pozadí, nebo se má ukončit.

Spouštění programů pomocí Watchera

Nechceme-li používat monitoring (nebo nám u konkrétní aplikace selhává – viz řešení problémů), můžeme použít sice trochu pracnější způsob spuštění aplikace, který je však použitelný pro všechny uživatele a nepotřebuje mít Watchera již spuštěného. Stačí vytvořit zástupce Watchera a v kombinaci s parametry /I, /N, /H, /R zadat ke spuštění danou aplikaci. Podporováno je i spouštění zástupců, souborů asociovaných s aplikacemi i aplikací se zadanými parametry.

Hlavní okno Watchera

Hlavní okno vyvoláme dvojklikem na tray ikonu, kliknutím pravým tlačítkem a výběrem volby *Aktivovat* nebo kliknutím při stisknuté klávese *SHIFT* nebo *CONTROL* (rychlý mód). Okno je rozděleno na několik panelů, které mohou být v závislosti na uživatelském nastavení skryty. K dispozici je všude kontextově citlivé menu, které aktivujete kliknutím pravým tlačítkem. Menu umožňuje vyvolání všech dostupných funkcí (což oceníme zvláště skryjeme-li panel tlačítek, která většinu funkcí obsluhují). Některé funkce jsou k dispozici jen z tohoto menu. Většina ovládacích prvků je také opatřena hints (stručnou bublinovou nápovědou, která se na pár sekund objeví při chvilkovém pozdržení myšího kurzoru nad ovládacím prvkem). Podívejme se nyní podrobněji na jednotlivé panely a jejich funkce.



Seznam procesů

Tento panel obsahuje tabulku se seznamem aktivních procesů a ukazatel automatického občerstvování na pozadí. K procesu vybranému v tabulce se zobrazují na informačním panelu dostupné informace. Vybranému procesu lze nastavit prioritu pomocí kontextového menu nebo na informačním panelu.

Ukazatel automatického občerstvování je zašedlý, pokud není občerstvování aktivováno. Je-li aktivní, zobrazuje aktuální fázi občerstvovacího procesu: *modrá* indikuje počátek občerstvovacího cyklu, *tmavě červená* první fázi cyklu (načítá se seznam procesů a zjišťují se změny oproti minulému cyklu), *jasně červená* druhou fázi cyklu (je-li zapnuto získávání detailních informací na pozadí, zjišťují se v této fázi), *zelená* převod získaných dat do tabulky a *černá* klidový stav (prodleva mezi cykly, kdy občerstvovací thread je v klidu). Aktivaci/deaktivaci občerstvování můžeme jednoduše provést dvojklikem na tento ukazatel nebo po kliknutí pravým tlačítkem na ukazatel zaškrtnout/odškrtnout volbu *Aktivní*.

Statistiky

Panel statistik zobrazuje nejdůležitější statistiky o aktuálním stavu systému a jeho výkonnosti. Danou statistiku lze zapnout/vypnout dvojklikem na příslušný ukazatel nebo po kliknutí pravým tlačítkem na ukazatel zaškrtnout/odškrtnout volbu *Aktivní* kontextového menu. Vlastnosti ukazatele (frekvence obnovování, barevné nastavení) lze nastavit výběrem volby *Vlastnosti...* z kontextového menu (viz dialog *Vlastnosti ukazatele*).

Zátěž procesoru	aktuální zatížení procesoru – pouze přibližná hodnota
Počet virtuálních počítačů	počet virtuálních počítačů v systému
Počet threadů	momentální počet threadů v systému
Využití paměti	souborná statistika vyjadřující obsazení dostupné paměti
Fyzická paměť	použitá a dostupná paměť RAM
Odkládací soubor	využití a velikost odkládacího souboru
Alokováno a zamčeno	velikost alokované a z toho zamčené paměti
Velikost diskové cache	minimální, aktuální a maximální velikost diskové cache
Počet čtení za sekundu	počet žádostí o čtení za sekundu a dosažené maximum
Počet zápisů za sekundu	počet žádostí o zápis za sekundu a dosažené maximum
Počet přečtených KB	počet KB přečtených za sekundu
Počet zapsaných KB	počet KB zapsaných za sekundu
Nevyřízená data	velikost dat, které čekají v cache na vyřízení

Poznámka:
Operace s některými diskovými zařízeními (např. i s CD-ROM) nejsou řízeny systémem souborů Windows (VFAT), a nejsou tudíž zachyceny v příslušných statistikách o čtení či zápisu v souborovém systému.

Tlačítka

Panel tlačítek obsahuje pět tlačítek s příkazy, které lze vyvolat snadno i z kontextového menu.

Zzzzz...
Zavře hlavní okno a ponechá Watchera jako tray ikonu.
Ekvivalent z kontextového menu: *Minimalizovat*
Ekvivalentní klávesa: *ESC*

Konfigurace

Otevře dialog *Konfigurace*.
Ekvivalent z kontextového menu: *Příkazy/Konfigurace...*

Občerstvit vše
Obnoví celý seznam procesů a zneplatní veškeré nasbírané údaje (čili uvede Watchera do stejného stavu jako po aktivaci hlavního okna).
Ekvivalent z kontextového menu: *Příkazy/Občerstvit*

Nový proces
Otevře dialog *Nový proces*, v němž můžete spustit nový proces.
Ekvivalent z kontextového menu: *Příkazy/Nový proces...*

Ukončit proces
Pokusí se ukončit vybraný proces ze seznamu procesů. Uživatel je předem informován o možných negativních důsledcích takového násilného zákroku. Používejte tuto volbu jen selhalo-li ukončení programu normálně, nebo pomocí dialogu Windows, který se zobrazí po stisku *CTRL-ALT-DEL*, anebo máte-li důvodné podezření, že tento trojhmat by Windows položil (někdy jsou totiž Windows příliš ohleduplné při ukončování procesu a dojde k ukončení nejen zaseknutého procesu ale celých Windows).
Ekvivalent z kontextového menu: *Příkazy/Ukončit proces*

Informační panel

Informační panel slouží (jak jinak) k zobrazení informací o vybraném procesu. Snadno (a hlavně rychle) dostupné informace jsou zobrazeny v horní části panelu společně s prioritou procesu, kterou lze výběrem ze seznamu změnit. Při zaškrtnutí pole *Zobrazit detailní informace* se Watcher pokusí získat všechny možné informace o vybraném procesu a zobrazí je na několika záložkách v panelu detailních informací. Zjišťování detailních informací, zvláště nejsou-li už načteny třeba automatických občerstvováním na pozadí, může trvat docela dlouho (podle situace a počítače až půl minuty – tradičně rekordmanem je proces KERNEL32.DLL, jádro Windows, který vlastní řádově tisíce paměťových bloků) a je operací značně náročnou především na paměťového manažera Windows (to se může projevit rozsáhlým odkládáním na disk a znovunačítáním stránek, obsadil-li proces mnoho paměťových bloků), je proto implicitně vypnuto.

Detailní informace:
Souhrn
Záložka *Souhrn* shrnuje obsah ostatních záložek a zobrazuje údaje, které uživatele budou zajímat asi nejvíce, tedy kolik paměti vybraný proces spotřebuje. Vysvětlení k jednotlivým položkám není snad potřeba, úplně srozumitelné nemusí být snad jen údaje *Koeficient záběru paměti moduly* a *Přibližné celkové paměťové nároky*.

Koeficient záběru paměti moduly vyjadřuje kolik paměti potřebuje proces na uložení používaných modulů vzhledem k tomu, že se s nimi o tyto moduly dělí. Např. máme aktivní modul XXX.DLL, který používá deset procesů. Modul XXX.DLL je velký 300 KB. Protože se ale o něj dělí vybraný proces s dalšími devíti procesy, naroste koeficient jen o 30 KB (300 : 10). Tento údaj je zavádějící ve Windows NT, kde se procesy dělí jen o systémové moduly, ostatní moduly existují fyzicky v paměti ve více kopiích a každý proces používá svou vlastní kopii (proto také mají Windows NT větší nároky na paměť, ale jsou zase stabilnější).

Přibližné celkové paměťové nároky jsou pak součtem koeficientu záběru paměti moduly, velikostí soukromých modulů a alokované paměti. Vytvářejí tak poměrně ucelenou představu o paměťových

nárocích procesu. Opět je tento údaj zavádějící ve Windows NT.

Paměť
Záložka paměť obsahuje dvě tabulky – tabulku použitých paměťových hald a k vybrané haldě v druhé tabulce seznam paměťových bloků této haldy.

Moduly
Pouze seznam modulů použitých vybraným procesem – nic víc.

Thready
Informace o threadech vybraného procesu. Bohužel thready jsou značně anonymní objekty a nelze k nim získat mnoho informací.

Okna
Téměř každý proces vlastní nějaké okno (na tom jsou Windows postaveny). V tabulce jsou vypsána tzv. top-level windows (česky okna nejvyšší úrovně), tedy okna, která už žádné jiné okno nevlastní, naopak ony mohou vlastnit další dceřinná okna (child windows).
Top-level okna jsou tedy ta okna, mezi nimiž se nachází i hlavní okno aplikace. Určení tohoto hlavního okna je poměrně složitá záležitost, protože top-level okna jsou si rovnocenná, a tak občas nemusí být v tabulce procesů zobrazeno jako hlavní okno to okno, které za hlavní považuje uživatel (tímto se za tuto chybu omlouváme).

Kontextové menu

Podívejme se nyní na funkce kontextového menu. Kontextové menu můžete vyvolat kdekoli v okně Watchera kliknutím pravým tlačítkem. Menu se přizpůsobuje místu, kde bylo vyvoláno, a momentální situaci (proto někdy nejsou všechny položky viditelné nebo dostupné).

Priorita procesu
Tato položka má ještě několik podpoložek, značka u některé z nich označuje prioritu vybraného procesu, kterou můžeme změnit pomocí tohoto menu pouhým výběrem nové položky. Položka *Priorita procesu* je viditelná jen tehdy, je-li viditelný panel procesů a je-li vybrán aktivní proces, a není závislá na místě vyvolání menu (tzn. pokud máte zapnutý panel procesů a vybrán aktivní proces, bude při vyvolání menu vždy viditelná).

Aktivní
Zaškrtnutí u položky *Aktivní* indikuje aktivitu ukazatele, na nějž bylo kliknuto pravým tlačítkem myši. Výběrem této položky se neaktivní prvek stane aktivním a naopak. Stejného efektu lze docílit dvojklikem na ukazatel.

Vlastnosti...
Je-li u ukazatele, na nějž bylo kliknuto pravým tlačítkem, k dispozici dialog pro uživatelské nastavení, objeví se i tato položka. Jejím výběrem otevřete příslušný dialog pro změnu vlastností ukazatele.

Příkazy
Podpoložky položky *Příkazy* jsou ekvivalentní čtyřem menším tlačítkům na panelu tlačítek. Toto menu je

užitečné tehdy, máme-li skryt panel tlačítek. Při vyvolání menu nad záložkou detailních informací přibudou ještě položky *Občerstvit detailní informace* a *Občerstvit tuto záložku*, které slouží k částečné obnově detailních informací vybraného procesu.

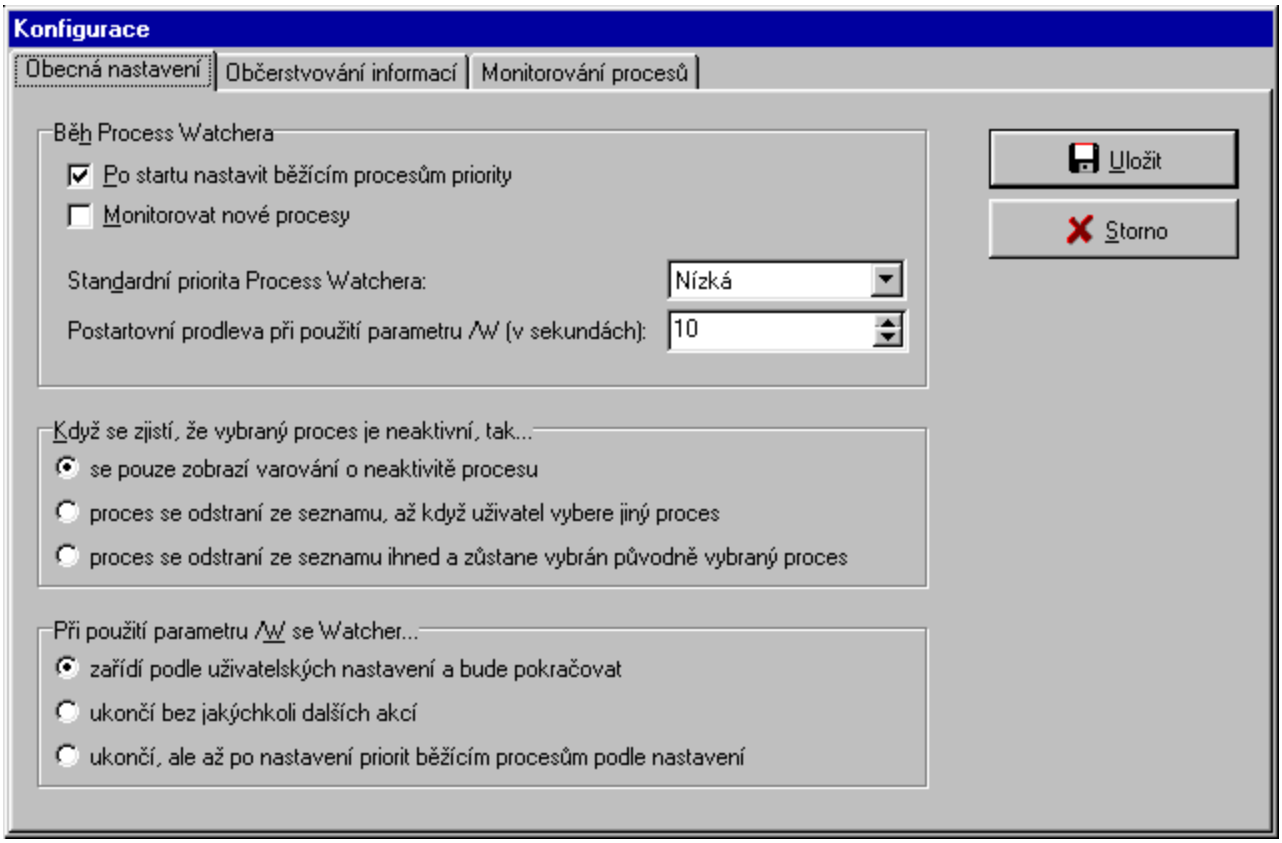
Zobrazit
Podmenu položky *Zobrazit* má dvě části oddělené čarou. V první části jsou zaškrtávací položky, kterými můžete zapnout/vypnout některý panel. Je-li zapnut informační panel, nelze vypnout panel seznamu procesů, je-li vypnut, je možné i měnit výšku hlavního okna. Nelze vypnout všechny panely. V druhé části podmenu jsou volby *Vždy navrchu*, díky níž můžete nechat hlavní okno nepřekryté jinými okny, i když budete pracovat s jiným programem, a *Zobrazit na poslední pozici* (dostupná je jen registrovaným uživatelům), při jejímž zaškrtnutí se při příštím vyvolání objeví hlavní okno ve stejných souřadnicích, kde bylo při svém zavření (jen v případě, že by nemělo být zobrazeno celé na ploše, bude posunuto); není-li tato volba zaškrtnuta, bude vždy okno vycentrováno.

O programu...
Ani Watcher nepostrádá toto nezbytné a populární dialogové okno.

Minimalizovat
Je to ekvivalent tlačítka *Zzzzz...*, které uzavírá hlavní okno a ponechává Watchera běžet jen jako tray ikonu v panelu úloh.

Konfigurace

Watcher disponuje širokými možnostmi konfigurace. Konfigurační dialog je rozčleněn na tři záložky: *Obecná nastavení, Občerstvování informací* a *Monitorování procesů*. Před prvním vyvoláním konfiguračního dialogu bude uživatel upozorněn, že tímto se zaregistruje (zde myšleno: vytvoří si konfigurační záznamy) a bude pak schopen využívat všech funkcí, které Watcher poskytuje. Všechny údaje zde nastavené jsou platné pouze pro aktuálního uživatele (v jednouživatelském režimu jsou pak platné pro všechny, kteří počítač používají). Stiskem tlačítka *Uložit* (nebo klávesy *ENTER*) aktuální nastavení v okně uložíte a uvedete je v platnost, bez uložení změn můžete dialog opustit stiskem tlačítka *Storno* (nebo klávesy *ESC*).
Doporučené nastavení je standardní, tedy to, které se objeví při prvním otevření dialogu *Konfigurace* (údaje na záložce *Monitorování procesů* je ovšem možno měnit dle libosti – i tu si dovolujeme nabídnout tip). Pokud měníte některá nastavení (týká se hlavně záložky *Občerstvování informací*), vyzkoušejte, zda nepůsobí potíže – např. na počítačích s menší pamětí doporučujeme nepoužívat získávání detailních informací na pozadí, které je paměťově někdy značně náročné.



Obecná nastavení

Běh Process Watchera
Po startu nastavit běžícím procesům priority je volba aktivující postartovní prohlídku systému, podle nastavení na záložce *Monitorování procesů* bude vybraným procesům změněna priorita.

Monitorovat nové procesy aktivuje možnost monitoringu, kdy jsou nově spouštěné procesy kontrolovány

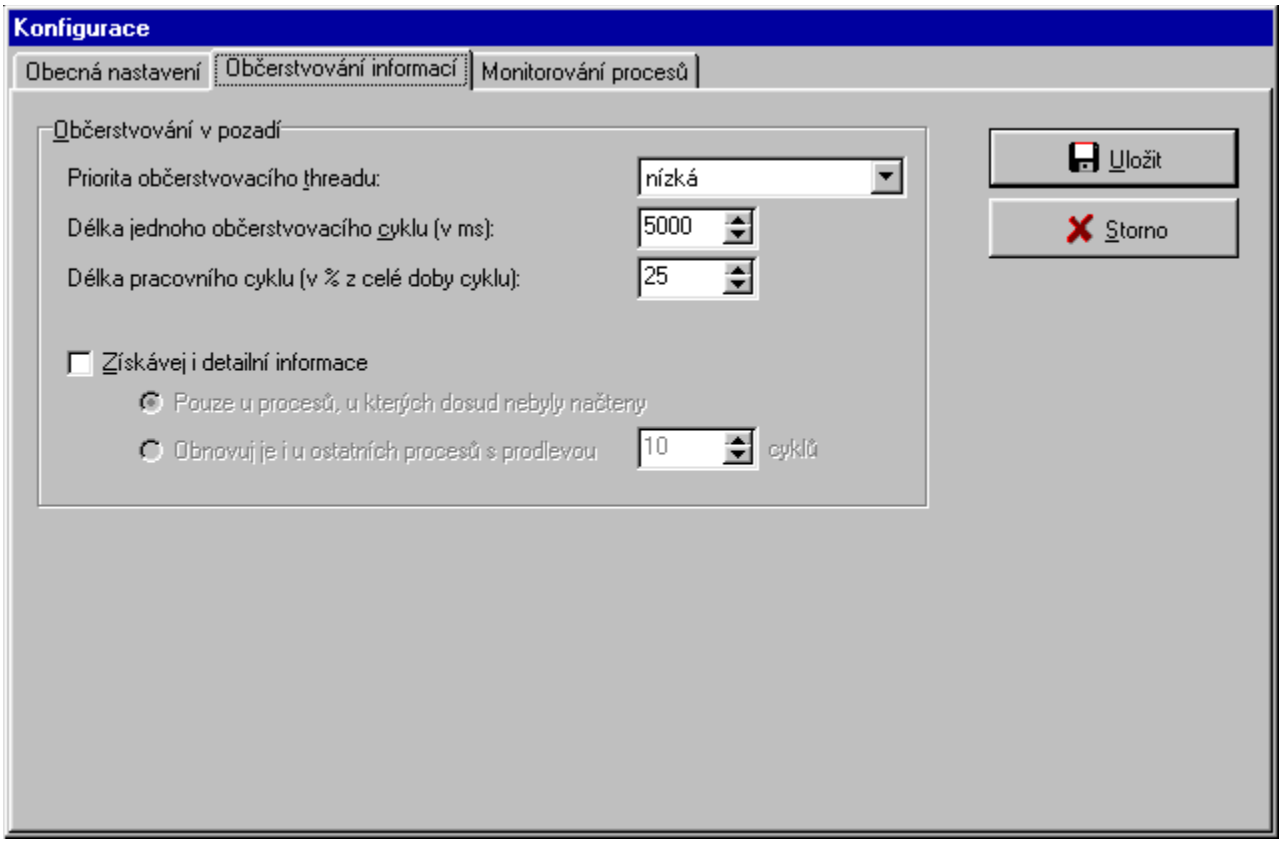
a v případě, že spouštěný proces je uveden v příslušném seznamu na záložce *Monitorování procesů*, je mu změněna priorita podle nastavení.

Standardní priorita Process Watchera určuje, jakou prioritu má mít Process Watcher, když běží v pozadí jako tray ikona v panelu úloh. Doporučená hodnota je *Nízká*.

Postartovní prodleva při použití parametru /W je doba, kterou má Watcher po spuštění s parametrem /W (používá se hlavně při startu Windows) vyčkat, než začne nastavovat běžícím procesům priority (je-li tato volba zapnutá). Při startu Windows se totiž spouští celá řada programů a mezi nimi i ty, které Watcher má kontrolovat. Pořadí spouštění určuje ale zavaděč Windows, a tak může být Watcher spuštěn dříve než programy, které má kontrolovat, může se tedy stát, že jim nebude moci přidělit správnou prioritu. Nastavíte-li dostatečně velkou prodlevu (standardní hodnota 10 sekund by měla být dostatečná), budou spuštěny všechny procesy, které mají být kontrolovány, ještě před započetím kontroly.

Když se zjistí, že vybraný proces je neaktivní, tak...
V tomto rámečku máte možnost ovlivnit, jak se změní seznam procesů při zjištění, že vybraný proces není již aktivní. První možnost zobrazí pouze varování, ale proces zůstane v seznamu, dokud se seznam občerstvením (ať již automatickým nebo ze strany uživatele příkazem) nezmění. Druhá možnost nechá uživatele se na proces podívat, ale varuje ho o neaktivitě procesu; poté, co uživatel vybere jiný proces, odstraní neaktivní proces ze seznamu. Poslední možnost odstraní neaktivní proces ze seznamu okamžitě, jakmile se uživatel pokusí jej vybrat. Pokud používáte automatické občerstvení, je doporučená první možnost, protože zbývající dvě narušují synchronizaci dat mezi občerstvovacím a hlavním threadem (a tak snižují výkon občerstvovacího threadu, protože je nutno v pozadí nasbíraná data zahazovat).

Při použití parametru /W se Watcher...
Pomocí tohoto nastavení máte možnost ovlivnit chování Watchera při použití parametru /W (typicky se používá při spouštění Watchera při startu Windows). První volba (obvyklá) ponechá Watchera aktivního po standardní inicializaci (tzn. Watcher zkontroluje běžící procesy podle nastavení, případně aktivuje monitoring a zůstane běžet jako tray ikona), druhá možnost jej ihned ukončí, třetí jej ukončí, ale až po kontrole běžících procesů.



Občerstvování informací

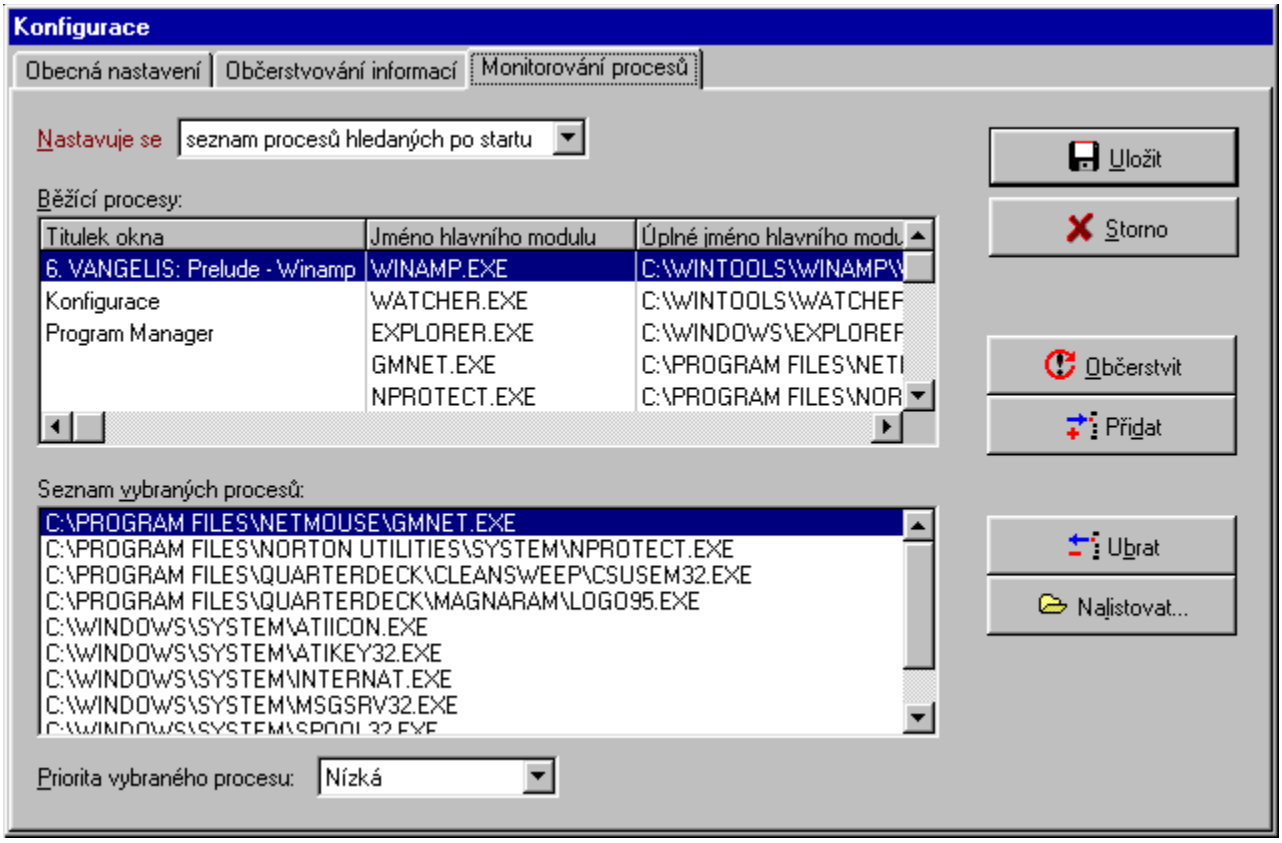
Pomocí záložky *Občerstvování informací* můžete nastavit vlastnosti automatického občerstvování. Automatické občerstvování probíhá na pozadí a uživatele nijak neruší. Jeho aktivita je signalizována pohybem ukazatele občerstvování na pozadí, který je v dolní části panelu seznamu procesů v hlavním okně. Co všechno přináší? Díky automatickému občerstvování je seznam běžících procesů stále aktuální, lze přednačítat detailní informace, takže jejich načtení pak netrvá tak dlouho (potřebný čas se zkrátí na cca 25 – 50 %), a aktualizovat je.

Priorita občerstvovacího threadu
Určuje, jakou prioritu má mít thread, který občerstvování provádí. Doporučená hodnota je *Nízká*.

Délka jednoho občerstvovacího cyklu
Občerstvování na pozadí probíhá v cyklech, jejichž průběh ukazuje ukazatel občerstvování v hlavním okně. Můžete nastavit délku cyklu. Délka cyklu určuje i aktuálnost seznamu procesů, čím kratší cyklus je, tím aktuálnější seznam je, ale tím větší nároky jsou kladeny na procesor. Doporučená délka cyklu je 5 000 milisekund.

Délka pracovního cyklu
Máte-li zapnuté získávání detailních informací (viz níže), můžete tímto ovlivnit, kolik času v cyklu má být věnováno na získávání detailních informací. První fáze cyklu (získání nového seznamu procesů) se provede nezávisle na délce pracovního cyklu, zbývá-li ještě nějaký čas do dosažení této hodnoty, je věnován druhé fázi cyklu (zisku detailních informací). Je-li tato hodnota příliš nízká a je malá i délka celého cyklu, nemusí se druhá fáze vůbec uskutečnit. Doporučená hodnota je 25 %.

Získávej i detailní informace
Zapnutím této volby aktivujete přednačítání detailních informací, jejich zobrazení pak nebude trvat tak dlouho, ale zvýší se dlouhoběji nároky na proces a na paměť. Volba má dvě možnosti:
Při výběru ***Pouze u procesů, u kterých dosud nebyly načteny*** se budou načítat detailní informace pouze u těch procesů, u kterých ještě nebyly načteny (ať manuálně nebo automaticky).
Při výběru ***Obnovuj je i u ostatních procesů s prodlevou xxx cyklů*** se přednostně načtou informace o procesech, u kterých ještě načteny nebyly, ale potom se každých ***xxx*** cyklů obnoví informace alespoň jednoho procesu.

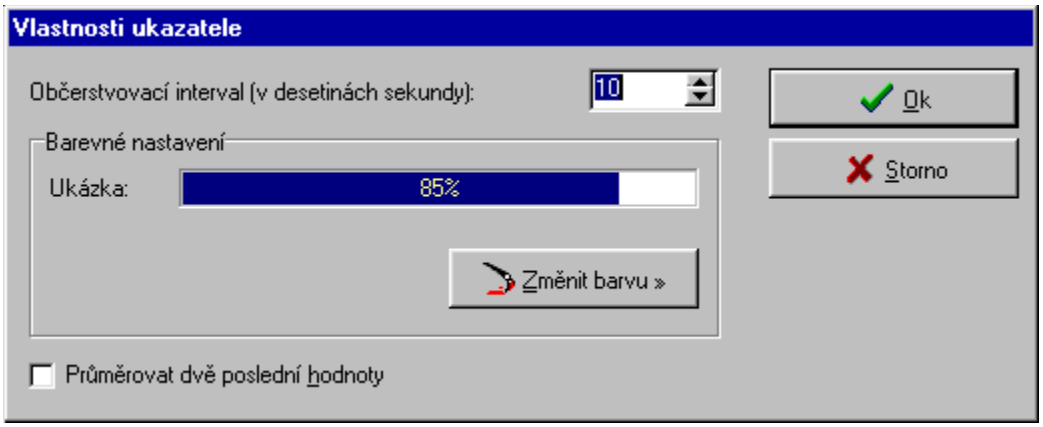


Monitorování procesů

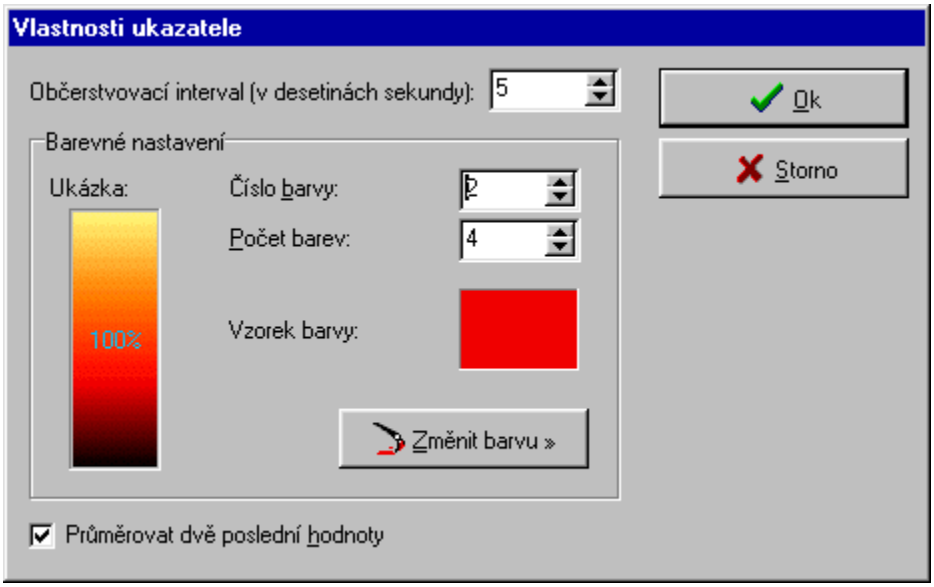
Záložka *Monitorování procesů* slouží k nastavení procesů, které mají být kontrolovány po startu Watchera, a procesů, které mají být sledovány při monitoringu. To, kterou skupinu procesů právě nastavujete, zjistíte podle seznamu *Nastavuje se* a můžete výběrem z něj změnit nastavovanou skupinu procesů. Tabulka *Běžící procesy* ukazuje, které procesy jsou aktivní. Tlačítkem *Přidat* (nebo stiskem klávesy *INSERT*) vybraný proces z této tabulky přesunete do seznamu vybraných procesů. Pokud se některý program, který chcete do seznamu přidat v tabulce nenachází, můžete jej nalistovat v dialogovém boxu, který vyvoláte stiskem tlačítka *Nalistovat...*; pamatujte, že je nutné zadat hlavní modul, nejste-li si jisti, raději aplikaci spusťte a přidejte z tabulky běžících procesů. Ze seznamu vybraných procesů můžete vysvícený proces odstranit stiskem tlačítka *Ubrat* (ekvivalentní je stisk klávesy *DELETE*). Tlačítkem *Občerstvit* aktualizujete tabulku běžících procesů.

Vlastnosti ukazatele

Existují dvě verze tohoto dialogu – jedna pro obyčejné ukazatele a druhá pro stínované ukazatele (tj. pro ukazatel zátěže procesoru). Obě verze mají společná tlačítka *Ok* pro uložení nastavení a *Storno* pro opuštění dialogu bez uložení změn, pole pro nastavení intervalu, ve kterém se bude ukazatel obnovovat, zaškrtávací pole *Průměrovat dvě poslední hodnoty*, jehož zatrhnutím bude ukazatel zobrazovat průměr poslední a aktuální hodnoty (takže nebude v případě velkých výkyvů tolik "poskakovat" – to je obzvláště užitečné u ukazatele zátěže procesoru), a dále modelový ukazatel, který demonstruje na náhodných hodnotách aktuální nastavení. Oba dialogy obsahují i tlačítko *Změnit barvu »*, po jehož stisku se objeví dvoupoložkové menu s položkami *Pozadí* a *Popředí*. Položka *Pozadí* mění v obou případech barvu podkladu ukazatele.



Verze dialogu pro obyčejné ukazatele
Položka menu *Popředí* mění barvu pruhu ukazatele.

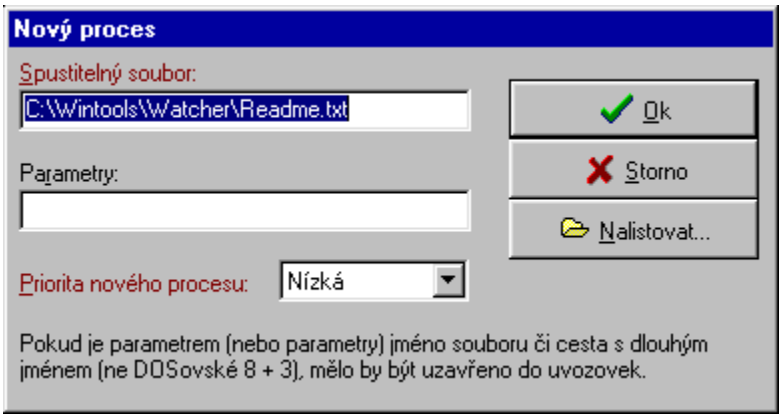


Verze dialogu pro stínované ukazatele
Verze pro stínované ukazatele obsahuje navíc pole pro nastavení počtu barev ve stínování, čísla barvy (barvy jsou číslované odshora dolů od nuly počínaje), která bude nastavovat při výběru položky menu *Popředí*, a barevného políčka se vzorkem nastavované barvy. Udělat několikanásobně stínovaný barevný ukazatel je hračkou: nastavíte počet barev a pak postupně zvolíte číslo barvy a volbou *Změnit barvu » / Popředí* ji nastavíte podle libosti.

Tip:
Kliknutím na modelový ukazatel zastavíte jeho pohyb na maximální hodnotě, takže budete vidět celé barevné spektrum, které vytváříte.

Nový proces

Funkce spuštění nového procesu z prostředí Watchera je jen doplňková, ale může se někdy hodit. Do pole *Spustitelný soubor* zapište jméno souboru, který se má spustit (může to být i dokument, zástupce nebo jiný soubor asociovaný s nějakou aplikací), nebo jej můžete nalistovat pomocí dialogového boxu, který vyvoláte tlačítkem *Nalistovat...*; je-li to nutné, do pole *Parametry* zapište parametry spouštěného souboru a v rozbalovacím seznamu *Priorita nového procesu* nastavte prioritu aplikace. Odklepnutím na tlačítko *Ok* soubor spustíte, tlačítkem *Storno* dialog opustíte. Registrovaným uživatelům se nastavení při stisku tlačítka *Ok* uloží a při příštím otevření dialogu *Nový proces* budou pole nastavena jako při posledním spouštění nového procesu.



Parametry Watchera

Parametry pro spouštění souboru

Syntaxe těchto parametrů: Watcher.EXE /X jméno_souboru [parametry_souboru]

/X je jeden z přepínačů */I, /N, /H, /R*. Za přepínačem následuje jméno souboru, který se má spustit, nepovinné jsou parametry souboru, které se uvádějí za jménem souboru. Pokud jméno souboru nebo parametry obsahují mezery, měly by být uzavřeny do uvozovek.
Při použití těchto přepínačů Watcher spustí zadaný soubor s danými parametry (jsou-li uvedeny) s prioritou podle přepínače: */I* = nízká (idle), */N* = normální (normal), */H* = vysoká (high) nebo */R* = realtimová (realtime).

Parametr /W

Syntaxe parametru /W je jednoduchá: Watcher.EXE /W

Watcher je spuštěn tak, že neregistrovaným uživatelům se ukončí, takže ani nepostřehnou jeho start, a u registrovaných uživatelů se zařídí podle nastavení z dialogu *Konfigurace* ze záložky *Obecná nastavení*. Tato volba se využívá hlavně při startu Windows. Zapnete-li pomocí Setupu start Watchera při startu Windows, je spouštěn právě s tímto parametrem.

How to...

Potřebujete zjistit rychle a snadno, jak udělat to či ono? Na základě připomínek testerů jsme připravili návod, jak zvládnout všechny běžné i méně běžné operace.

Tipy (nejen pro dnešek)

Pokud by Vás zajímalo několik tipů a doporučení, jste na správné stránce.

Doporučené nastavení priorit
Je dobré nechat Watchera spouštět při startu Windows a nechat jej nastavit priority spuštěných programů (pokud někomu vadí jeho přítomnost, může ho nechat pomocí nastavení parametru /W v dialogu *Konfigurace* na záložce *Obecná nastavení* po spuštění a nastavení priorit ukončit).

Vřele doporučuji, abyste nechali nastavit nízkou prioritu všem aplikacím, které nejsou nezbytně nutné (pokud něco takového vůbec používáte) a které jen slouží ke zvýšení komfortu uživatele (a požírání systémových zdrojů). Mezi tyto aplikace počítáme různé panely nástrojů, plovoucí okna s poznámkami a další "pomocníky", kteří většinu času stráví čekáním na vyvolání. Některé si nízkou prioritu nenastavují (asi se považují za nepostradatelné), a tak brzdí procesy, které nepostradatelné jsou. Jestliže se bez těchto brzd systému opravdu neobejdete (ne všechny takové nástroje musí být špatné, některé naopak mohou být velice povedené), zkuste jim alespoň nastavit nízkou prioritu, odlehčíte tím ostatním programům, které pak poběží rychleji. Vše ale něco stojí, a proto musíte počítat s tím, že programy s nízkou prioritou budou reagovat pomaleji, zvláště poběží-li jiná náročná aplikace.

Je možné si trochu pohrát i s prioritami některých systémových programů. Otestované nastavení, jak se autorovi osvědčilo, je v tabulce:

Modul	Priorita	Funkce modulu
MSGSRV32.DLL	vysoká	rozesílá zprávy všem aplikacím, stará se o frontu zpráv
SPOOL32.EXE	nízká/normální	spooler tisku
INTERNAT.EXE	nízká	přepínač klávesnice
SYSTRAY.EXE	nízká	program spravující oblast tray ikon na hlavním panelu

Poznámka:
Nastavení SPOOL32.EXE je nutné vyzkoušet – záleží velice na tiskárně. Nemáte-li tiskárnu, můžete jej bez problémů nastavit na nízkou prioritu, v opačném případě zkuste experimentovat. Výkonné tiskárny vybavené velkým bufferem a výkonným procesorem, stejně jako hlemýždí tiskárny jehličkové nebudou zřejmě vyžadovat rychlou odezvu počítače (můžete zkusit nízkou prioritu), naopak u jiných tiskáren se musí počítač po celou dobu tisku o tisk starat (bude lepší nechat normální prioritu).
Zvýšením priority MSGSRV32.EXE byste měli docílit rychlejší distribuce zpráv Windows jednotlivým oknům, a tedy i o něco rychlejší odezvy (myšleno, že nemusíte tak dlouho čekat, než se po kliknutí objeví menu, než se překreslí okno atd., pokud program něco počítá, nemá to se zprávami a prioritou tohoto modulu co dělat).

Neměňte prioritu modulu KERNEL32.DLL! KERNEL32.DLL je modul jádra Windows, snížením jeho priority Windows určitě neprospějete: když jádro (které musí vyřizovat velké množství požadavků ostatních procesů a má řídící úlohu) bude postaveno na stejnou úroveň jako ostatní programy, dojde nejvýš ke zpomalení systému. Ani opačná cesta není dobrá: přidělíte-li jádru realtimovou prioritu, je to absolutně spolehlivý způsob, jak nechat Windows zamrznout, a to tak důkladně a naráz, že doslova v okamžiku strnou a pomůže jen tvrdý reset. Ani ukončení procesu jádra není dobrý nápad (co se stane si každý asi domyslí).

Realtimovou prioritu používejte jen výjimečně a máte-li k ní důvod! Zkusíte-li nastavit nějakému procesu realtimovou prioritu, je to risk, a riskujete tím více, čím náročnější ten proces je a čím pomalejší

počítač máte. Proces s realtimovou prioritou (to vyšší priorita než jakou má jádro!) si vyžádá tolik času na úkor ostatních procesů, že systém nemá čas ani na základní operace (jako je snímání hardwarových událostí, generování příslušných zpráv a jejich rozesílání jednotlivým oknům jejich překreslování atd.) – tj. Windows vypadají jako v posledním tažení, těsně před pádem, který se ne a ne dostavit (myš ještě chaoticky skáče po obrazovce, okna se překreslují jak ve zpomaleném filmu a na reakci na kliknutí myší nebo stisk klávesy si uživatel může pěkně dlouho počkat). Toto je katastrofický scénář, u většiny programů k tomuto nedojde.
Pokud k tomu ale náhodou dojde, nemačkejte chaoticky tlačítka myši ani klávesnice, nehýbejte zbytečně myší, máte ještě šanci vyhnout se resetu nebo trojhmatu. Pomalu, trpělivě a soustředěně posunujte myš k ikoně Process Watchera, jednou na ni klikněte a držte klávesu *CTRL* nebo *SHIFT* (abyste aktivovali rychlý mód, pokud máte nastaveno občerstvení v pozadí na trochu náročnější parametry, situace by se mohla ještě zhoršit) dokud se neobjeví okno Watchera. Opět opatrně a soustředěně myší (nebo možná raději klávesnicí, pokud znáte klávesy pro pohyb ve Windows) vysviťte problematický proces a změňte mu prioritu na normální nebo i nízkou. V tomto okamžiku by Windows měly chytit dech a začít chovat se normálně. Pamatujte, že musíte být velice, velice trpěliví, jinak se minete úspěchem!
Pokud si nějaká aplikace sama od sebe nastaví realtimovou prioritu, asi k tomu má důvod a neměli byste ji nijak měnit (příkladem je např. modul DDHELP.EXE, což je pomocný proces pro rozhraní DirectX, ten má pro realtimovou prioritu důvod a jeho priorita ani nepůsobí potíže).

Nesnažte se zvyšovat prioritu všem procesům.
Ničeho tím nedosáhnete, nanejvýš paradoxně zpomalení systému. Priorita určuje relativní podíl procesorového času, který je věnován danému procesu. Mají-li všechny procesy stejnou prioritu, je jim přiděleno stejně času, tj. všichni mají stejně, a běží stejně rychle. Efektivnější je snižování priorit méně důležitým procesům, které pak přestanou brzdit ostatní procesy.

Opatrně s Win16 procesy.
Win16 procesy jsou běžící programy určené původně pro Windows nižších verzí (tj. Windows 3.x, popřípadě nižším verzím Windows NT, nebo hybridním sestavám Windows 3.x rozšířené o knihovnu Win32s). Už z toho vyplývá, že se chovají úplně jinak než řádné aplikace pro Windows 95, Windows 98 a novější verze Windows NT (tzv. Win32 aplikace). Mají vlastní adresový prostor, vlastní jádro a systémové moduly a vlastně všechny dohromady tak trochu běží jako jeden Win32 proces, aby bylo možné emulovat kooperativní multitasking z Windows 3.x – výsledkem je, že zásahy do priorit Win16 procesů zřejmě příliš neovlivní Win32 procesy (ty jsou ovlivněny už pouhou skutečností, že Win16 proces v systému existuje), ale může způsobit nerovnováhu mezi Win16 procesy. Těžko předpovědět, jak se v konečném důsledku takový zásah projeví.

Kdy nepoužívat monitoring?
Monitoring je spíše experimentální funkce, a ač již byl mírně zdokonalen a bylo věnováno velké úsilí, aby nepůsobil problémy, stále se problémy mohou objevovat. Proto používejte monitoring pouze tehdy, nebude-li to působit problémy jiným aplikacím (zatím nepříliš dobře prozkoumané a zdůvodněné příčiny mohou způsobit pád některé aplikace nebo Watchera), nebude-li Vám to život spíše komplikovat a používáte-li především obvyklý software. Monitoring si tedy před dlouhodobějším používáním dobře otestujte a používejte jej, jen pokud se osvědčí. Ukázalo se také, že jeho stabilita je závislá na různých faktorech, které jsou dány konkrétním počítačem. Do té doby, než se podaří najít lepší techniku pro monitorování, nelze jej masově doporučit.

Známé problémy

Zjistíte-li řešení některého problému, dejte nám vědět. Pro odborníky je u některých problému kousek zdrojového kódu, který je podezřelý z nesprávného chování (kód je v Delphi 3).

Chyba Setupu ve Windows NT je značně pravděpodobná (podle dostupných materiálů), Setup totiž zřejmě nebude schopen správně zlikvidovat všechny registrační záznamy a bude nutno jej odinstalovat ručně. V případě, že by se naše podezření potvrdilo, byli bychom vděčni za spolupráci s někým, kdo má nainstalovány Windows NT a je ochoten zahrát si na pokusného králíka, abychom dokázali tuto pravděpodobnou chybu odstranit.

Další pravděpodobné problémy ve Windows NT by mohly nastat vzhledem k poměrně silnému důrazu Windows NT na bezpečnost, kdy by mohl někde (pravděpodobně také při instalaci nebo odinstalaci) Watcher také narazit – např. nebude moci odstranit některé zástupce či dostat se k registračním záznamům všech uživatelů, protože jeho uživatel nebude mít dostatečnou úroveň oprávnění pro takovou operaci. Tuto hypotézu jsme také nemohli ověřit a budeme rádi, pokud někdo na tuto chybu narazí a oznámí nám ji.

Dlouhá prodleva mezi možností opět vyvolat hlavní okno po jeho uzavření je způsobena nutností uvolnit nasbírané informace o procesech. Pokud je těchto informací velké množství (zvláště je-li zapnuto získávání detailních informací na pozadí), může opravdu tento proces trvat dosti dlouho. S tím se ale těžko dá něco dělat, ale ukáže-li se to vážným problémem, zkusíme změnit strukturu programu, aby uživatele tento problém tolik netížil.

Přerovnávání procesů v seznamu procesů je normální jev při zapnutém automatickém občerstvování; je však uděláno tak, že ač se změní pořadí procesů, nebo jsou některé položky ze seznamu vypuštěny nebo naopak přidány, vysvícen zůstává stále stejný proces. Bylo by asi nepříjemné, kdybyste si prohlíželi detailní informace k některému procesu a on náhle z tabulky zmizel, nebo by se přemístil a místo by se vybral jiný proces. Toto chování je nezávislé na nastavení varování/odstraňování procesů ze seznamu, které se nastavuje v dialogu *Konfigurace*; to se týká jen výběru uživatele.

Není-li uveden titulek hlavního okna v seznamu procesů může to znamenat, že hlavní okno nemá (resp. nepodařilo se ho najít, či je to okno skryté), nebo že je aplikace minimalizována (tj. všechna okna jsou skrytá). Řiďte se jménem modulu (nebo celou cestou), popřípadě si můžete nechat zobrazit detailní informace a na záložce *Okna* z titulků oken zjistíte, o kterou aplikaci jde. Je-li aplikace minimalizovaná, můžete ji též obnovit z minimalizovaného stavu a občerstvit seznam procesů (nebo počkat až se aktualizuje sám, máte-li zapnuto automatické občerstvování), pak se už v seznamu objeví její hlavní okno.

DOSovský program není v seznamu uveden. Tedy on tam uveden je, ale jsou tu dva problémy. První problém je v tom, že neběží-li DOSovský program v okně, je jeho okno skryto a dostáváme se k předchozímu problému. Narozdíl od Windows aplikací však u DOSovských programů nesouhlasí jméno modulu (natožpak celá cesta) se jménem spuštěného souboru. To je proto, že v prostředí Windows se musí i DOSovský program chovat jako Windows aplikace, což je zaručeno pomocným programem, v jehož rámci DOSovský program běží (ve Windows 95 je tímto pomocným modulem modul WINOA386.MOD). Samozřejmě, že informace o takovémto DOSovském procesu jsou značně zkreslené.

Nejsou přesné údaje u aplikací pro Windows nižších verzí jako třeba pro Windows 3.1. Důvod je obdobný jako u DOSovských programů. Šestnáctibitové programy pro Windows (Win16 aplikace) běží ve zvláštním módu, kdy se emuluje chování 16bitových Windows v prostředí 32bitovém; mají vlastní

adresový prostor a vlastní systémové moduly. Rutiny s nimiž Watcher pracuje jsou funkce 32bitového jádra, které fungují (jakžtakž) přesně jen u plnohodnotných 32bitových aplikací pro 32bitové Windows (Win32 aplikace).

U některých programů nefunguje monitoring. I to se může stát v několika případech. První případ: program se skládá z více programových modulů a program, který se má monitorovat je vlastně jen zavaděč, který spustí hlavní modul, u nějž by monitorování smysl mělo. Druhý případ: program jej DOSovskou aplikací – je to vlastně speciální případ první možnosti, kdy zavaděčem je výše zmiňovaný pomocný modul, v jehož rámci DOSovský proces běží; vlastní spustitelný soubor pro DOS v podstatě není procesem Windows a nelze k němu najít potřebné údaje, aby jej monitoring vyhodnotil jako proces, kterému se má změnit priorita. Ostatně u DOSovských programů nemá ani moc cenu prioritu měnit a vůbec nejlepší je, zvláště v případě, že jsou to náročné programy (jako hry), je spustit v režimu MS-DOS. Třetí případ je dosti kuriózní, ale vyskytnout se může: program nevytvoří vůbec žádné okno – to znamená, že to asi bude systémová aplikace (a to pak není radno ji prioritu nějak moc měnit), protože až na systémové moduly (např. KERNEL32.DLL) snad každý program běžící pod Windows vytvoří alespoň jedno okno pro zachytávání zpráv (zda má okna si můžeme snadno ověřit na záložce *Okna* při zapnutých detailních informacích). Proces, který nevytvoří alespoň jedno okno, nedokáže monitoring zachytit. Problematika zjištění spuštění aplikace je diskutována v části pro odborníky.
Monitoring může mít i další negativní důsledky, které nebyly ještě spolehlivě zdokumentovány; z testů ale vyplynulo, že reakce aplikací a systému na monitoring je značně různorodá a liší se případ od případu – je totiž dána mnoha okolnostmi, především kondicí Windows a provozovaným softwarem. Obecně se zdá, že Windows s Internet Explorerem jsou méně stabilní a náchylnější k různým těžko odhalitelným chybám nebo chybám, které lze jinak obtížně vysvětlit. Na počítači, kde byl Watcher vyvíjen (Windows 95 OSR 1, tj. verze 4.00.950, bez Internet Exploreru, sítě, s omezenou instalací MS Office 95 – bez panelu zástupců, rychlého startu a rejstříku dokumentů), byl Watcher v koncových verzích velmi stabilní a nezpůsobil zhroucení žádné aplikace ani se sám nezhroutil.

Některé moduly, ač jsou hlavními moduly procesu, nejsou vůbec v seznamu modulů a dokonce takový proces nemusí mít žádné soukromé moduly (ze standardních např. MMTASK.TSK, MSGSRV32.EXE). Nevíme, čím je toto způsobeno; pravděpodobně se jedná o určitou abnormalitu vyplývající z určitého výsadního postavení těchto procesů (jedná se o systémové nebo pomocné moduly). U normálních aplikací tento jev ale pozorován nebyl.

Při zobrazení některého dialogu (např. Konfigurace) se dočasně zruší nastavení *vždy na vrchu*, protože jinak byly problémy se standardními dialogovými boxy pro nalezení souboru a nastavení barvy – po vyvolání se totiž ocitly pod dialogovými okny, která je vyvolala, a nebylo možno k nim přistupovat, nebo se dokonce nezobrazily vůbec (ač přijímaly stisky kláves a očividně byly aktivní!). Nejschůdnějším řešením bylo vždy před zobrazením modálního okna zrušit dočasně nastavení *vždy navrchu*, po dobu zobrazení těchto modálních oken tedy Watcher navrchu není.

Menu spojené s tray ikonou nemizí ihned po kliknutí mimo něj ale až po přejetí myší přes něj. Příčina je neznámá, ale ostatní programy, sídlící v panelu úloh se s tímto problémem nepotýkají.

Zkrácený okomentovaný zdrojový text z wndproc obsluhující zprávy hlavního okna Watchera:

```
function MainWndProc(Wnd: HWND; Msg: UINT; WParam: WPARAM; LParam: LPARAM): LRESULT; stdcall;
var
 APoint: TPoint;

begin
 Result := _NULL;

 case Msg of
```

```
  WM_DESTROY  : PostQuitMessage(_NULL);

  // zpráva z panelu úloh, že se děje něco s mou tray ikonou
  WM_TRAYICON : case LParam  of
                // levé tlačítko bylo puštěno nad tray ikonou a právě se drží Shift nebo Control
- aktivuje se hlavní okno Watchera v rychlém módu
                // bohužel musí být použito GetAsyncKeyState místo GetKeyState, protože jiné
zprávy (jako stisky kláves) hlavní okno nepřijímá,
                // neboť je neviditelné, a tak při použití GetKeyState docházelo při aktivaci
Watchera k náhodné aktivaci rychlého módu
                WM_LBUTTONUP     : if (GetAsyncKeyState(VK_SHIFT) < 0) or
(GetAsyncKeyState(VK_CONTROL) < 0) then DoActivateWatcher(true);
                // v případě dvojkliku se aktivuje hlavní okno Watchera v normálním módu
                WM_LBUTTONDBLCLK : DoActivateWatcher(false);
                // při puštění pravého tlačítka nad tray ikonou, musíme vyvolat popup menu, zde
je asi problém
                WM_RBUTTONUP     : begin
                                     GetCursorPos(APoint); // získáme souřadnice myšího kurzoru
                                     with APoint do
                                      // a na těchto souřadnicích necháme vyvolat popup menu;
zkoušel jsem i TrackPopupMenu s druhým parametrem
                                       // TPM_BOTTOMALIGN or TPM_RIGHTALIGN or TPM_RETURNCMD
(vrací přímo ID příkazu) se stejným výsledkem
                                       TrackPopupMenu(QuickMenu, TPM_BOTTOMALIGN or
TPM_RIGHTALIGN, X, Y, _NULL, MainWindow, nil);
                                     end;
               end;

  // zpracujeme příkazy popup menu
  WM_COMMAND  : case SmallInt(WParam) of
                 midActivate   : DoActivateWatcher(false);
                 midExit       : DestroyWindow(MainWindow);
                 midDisableMon : ...
```

Titulek hlavního okna v seznamu procesů neodpovídá hlavnímu oknu – to může být poměrně častý jev, se kterým se ale dá těžko něco dělat. Okna ve Windows jsou totiž více méně nezávislá a rovnocenná. Má-li program více top-level (čili oken nejvrchnější úrovně, oken, která nejsou vlastněna jinými okna, naopak sama mohou vlastnit dceřinná okna), je těžké určit, které okno je oknem hlavním. Někdy je to ještě komplikovanější o to, že okno, které za hlavní považuje uživatel není totožné se skutečným hlavním (aplikačním) oknem (to je už otázka programátorského řešení, např. i Watcher má aplikační okno celou dobu běhu skryté a okna, se kterými pracuje uživatel nejsou hlavními okny, ač se tak uživateli jeví). Protože se tedy jednoznačně nedá určit hlavní okno (a někdy by to ani k ničemu nebylo, protože hlavní okno je skryto), používá Watcher řadu podmínek, jimiž postupně vyloučí okna, která pravděpodobně uživatel za hlavní okna považovat nebude. Podmínky pro hlavní okno jsou: okno je viditelné a aktivní, má vůbec nějaký popisek a je umístěno tak, že je alespoň jeden bod v prostoru plochy. Vyhovuje-li více takových oken, vybere se okno s nejmenším handle, tedy to, které bylo pravděpodobně vytvořeno nejdříve.

Rutina získávající hlavní okno procesu:

```
// typ pro předávání dat callback funkci
type
 PGPMWData = ^TGPMWData;
 TGPMWData = record
  ErrorFree: Boolean;    // indikace hlavní funkci, zda callback proběhl bez výjimky
  ProcessID: Integer;    // ID procesu, ke kterému má být nalezeno hlavní okno
  ScreenRect: TRect;     // souřadnice plochy - chceme, aby hlavní okno leželo na ploše a ne
někde mimo, aby nebylo vidět
  MainWindow: HWND;      // výsledek - handle hlavního okna, v případě neúspěchu rovno NULL
 end;

// pomocná funkce - callback pro Windows; první parametr je handle zkoumaného okna, druhý je
ukazatel na strukturu TGPMWData
```

```
function _GetProcessMainWindow(Handle: HWND; Data: PGPMWData): Boolean; stdcall;
// pomocné proměnné
var
 Dummy: Integer;
 wndRect: TRect;

begin
 // pokud se nic nepřihodí, bude callback pokračovat do vyčerpání top-level oken
 Result := true;
 try
  // jestliže je to opravud okno, je viditelné a aktivní
  if IsWindow(Handle) and IsWindowVisible(Handle) and IsWindowEnabled(Handle) then
   begin
    // získáme process ID okna
    GetWindowThreadProcessID(Handle, @Dummy);
    // jestliže okno patří ke zkoumanému procesu...
    if Dummy = Data^.ProcessID then
     begin
      // ... a má nějaký titulek...
      if GetWindowTextLength(Handle) > 0 then
       begin
        // zjistíme, jak se protíná se souřadnicemi plochy
        SetRectEmpty(wndRect);
        GetWindowRect(Handle, wndRect);
        if IntersectRect(wndRect, wndRect, Data^.ScreenRect) then
         // leží-li alespoň částí v ploše, pak vezmeme za výsledek nejmenší dosud získané handle
- toto okno bylo (asi) vytvořeno dříve
         with Data^ do
          if (MainWindow = _NULL) or (Handle < MainWindow) then MainWindow := Handle;
       end;
     end;
   end;
 except
  // nastala-li jakákoli výjimka, označíme ji ve výsledku a ukončíme callback
  Data^.ErrorFree := false;
  Result := false;
 end;
end;

// hlavní funkce; parametrem je ID procesu, k němuž chceme získat hlavní okno, návratovou
hodnotou handle tohoto okna (v případě neúspěchu NULL)
function GetProcessMainWindow(ProcessID: Integer): HWND;
// data pro callback
var
 CallbackData: TGPMWData;

begin
 // naplníme data pro callback
 CallbackData.ProcessID := ProcessID;
 with CallbackData do
  begin
   ErrorFree := true;
   MainWindow := _NULL;
   SetRectEmpty(ScreenRect);
   ScreenRect.Right := GetSystemMetrics(SM_CXSCREEN);
   ScreenRect.Bottom := GetSystemMetrics(SM_CYSCREEN);
  end;

 // necháme Windows, aby prošly seznam top-level oken
 EnumWindows(@_GetProcessMainWindow, Integer(@CallbackData));
 // došlo-li k chybě, vyvoláme výjimku
 if not CallbackData.ErrorFree then raise EGSWalkError.Create(SWindowsEnum);
 // jinak vrátíme výsledek
 Result := CallbackData.MainWindow;
end;
```

Řešení problémů

Pokud jste se setkali s problémem, doufáme, že Vám budeme schopni pomoci a že tu najdete odpověď na otázku *Proč to (zase) nejde?*

? Zapnul jsem si detailní informace k jednomu procesu a Watcher se zhroutil.

! Tento problém nás náhodou také při testování jednou potkal. Výjimečně k této chybě může dojít. Abyste se neděsili, není to nic závažného, protože půjde pravděpodobně o jednorázovou chybu Windows. Zkuste Windows restartovat. Pokud to nepomůže a Watcher se bude hroutit stále, půjde asi o chybu závažnějšího charakteru skrytou někde uvnitř Windows (myšleno konkrétně Vaší instalace), se kterou my těžko něco uděláme. Zkuste Watchera u kamaráda nebo známého, který má stejnou verzi Windows. Pokud se chyba bude opakovat, není s největší pravděpodobností Watcher zcela kompatibilní s touto verzí. V tomto případě nás informujte o všech okolnostech (který program působí hroucení, jaká hláška se zobrazí, jakou verzi Windows máte...).

? Zkusil jsem dát jednomu programu realtimovou prioritu a Windows se mi zbláznily.

! Opatrně s realtimovou prioritou – viz problematika používání realtimové priority z kapitoly *Tipy (nejen pro dnešek)*.

? Od instalace Watchera se mi pravidelně hroutí některé programy, které před tím fungovaly bezvadně.

! Tato skutečnost může být způsobena monitoringem. Zkuste jej vypnout (nejlépe pomocí dialogu *Konfigurace*, nebo alespoň pomocí menu u tray ikony) a spustit problematickou aplikaci znova. Pokud se opět zhroutí, vypněte monitoring v každém případě pomocí dialogu *Konfigurace* a Windows restartujte. Zhroutí-li se aplikace znovu, nebude vina na straně Watchera. Bude-li dlouhodobě korektně fungovat, zřejmě je hroucení reakce na monitoring, takže jej raději nepoužívejte.

Slovníček pojmů a vysvětlivek

Některé pojmy a obraty použité v této nápovědě se týkají pouze Watchera, jejich rejstřík je uveden tady:

monitoring
rychlý mód

Kromě pojmů týkajících se pouze Watchera, je v nápovědě zmíněno poměrně mnoho obecných termínů. Aby nevzniklo nedorozumění způsobené určitou nejednotností terminologie, a také pro méně znalé uživatele, uvádíme rejstřík termínů v nápovědě použitých:

adresový prostor
always on top
buffer
cache
dialog
freeware, shareware, public domain
halda a paměťový blok
handle
hint
hook
chráněný mód
mapping (mapování)
mód virtuální 8086
modul
multitasking
multithreading
okno
paměť
patch
priorita
proces
reálný mód
Registry
systémové zdroje
swapfile (odkládací soubor)
thread
tray icon
virtuální počítač
Win16
Win32
window message (zpráva okna)
window procedure (wndproc)

Monitoring

Monitoring je funkce Watchera, která umožňuje sledovat vznik nových procesů a upravovat jejich prioritu hned po jejich vzniku.

Rychlý mód

Rychlý mód je mód, v němž Watcher vypíná automatické občerstvování na pozadí, aby byl rychlejší a nebrzdil systém. Používejte při velkém zatížení systému. Aktivuje se (jednoduchým) kliknutím na tray ikonu za stisknuté klávesy *SHIFT* nebo *CONTROL* (tu držte, dokud se neobjeví okno Watchera). Rychlý mód je indikován v titulku hlavního okna.

Adresový prostor

Každý proces je schopen operovat s adresami paměti (tj. adresovat paměť) v určitém rozsahu, který je dán architekturou operačního systému a procesoru. Oblast paměti, kterou je proces schopen adresovat se nazývá adresový prostor.
Virtuální adresový prostor je takový adresový prostor, který není absolutně vztažen do globálního adresového prostoru, může se nacházet kdekoli v paměti, adresy procesu tedy jsou platné pouze v rámci jeho virtuálního adresového prostoru. Ve Win32 může být tento adresový prostor velký až 2 GB a je dán 32bitovou strukturou systému.

Always on top

Always on top (čili česky vždy navrchu) je nastavení okna, kdy není překrýváno normálními okny. Tato okna tedy jsou vidět vždy; takováto okna se ale vůči sobě chovají jako rovnocenná normální okna a mohou se překrývat, vidět celé je pak aktivní okno.

Buffer

Buffer je obecný název pro vyrovnávací paměť. Vyrovnávací paměť se používá při komunikaci mezi různě rychlými zařízeními (např. paměť – disk), kdy se do vyrovnávací paměti uloží data, která jsou později postupně zpracována. Výsledkem je, že pomalejší zařízení má dostatek času na zpracování dat a rychlejší na něj nemusí čekat.

Cache

Cache je vlastně buffer, vyrovnávací paměť. Za cache se ale většinou označuje inteligentní buffer, tj. buffer, který se řídí určitými algoritmy pro výběr bufferovaných dat. Nejčastější případ cache je disková cache. Systém cache Windows (VCACHE) plní hned dvě úlohy: cachuje (ošklivé ale těžce opsatelné slovo) FAT tabulku (tj. tabulku s pozicemi souborů na disku), pamatuje si tedy, pro který soubor kam sáhnout na disk, a pak slouží jako buffer při souborových operacích – při zápisu na disk, jsou data uložena do cache a pak postupně zapisována na disk, jak to jen zátěž systému a disku dovolí, a při čtení z disku je soubor přednačítán do cache, a je-li požadován již načtený blok souboru, načítá se z cache (tím se zkracuje přístupová doba k disku a snižuje počet nutných čtení, protože se soubor načítá méněkrát). Příkladem využití cache, který je dobře demonstrovatelný, je kopírování na disketu: zkuste nějaký větší soubor (aspoň 1 MB) kopírovat ve Windows na disketu – ukazatel naběhne do plné pozice, jak je soubor načten do cache, a pak už se jen čeká na dokončení zápisu; když pak hned soubor z diskety zkusíte zkopírovat zpět na disk (třeba do jiné složky), proběhne kopírování podezřele rychle, a to proto, že se soubor nečte z diskety, ale z cache.

Dialog (dialogové okno)

Dialog je typ okna, které většinou nemá libovolně měnitelnou velikost, a je v modálním stavu, tj. uživatel nemůže pracovat s programem, který tento dialog vyvolal, dokud dialog neukončí. Dialog se toto okno nazývá proto, že v něm téměř vždy probíhá činnost, kdy program vyžaduje od uživatele instrukce jako odpověď na nějaký příkaz (tedy probíhá dialog program – uživatel).

Freeware, shareware, public domain, firmware

Všechna tři slova v titulku jsou označení pro různé typy software.

Shareware je software, který sice smí být šířen volně, ale uživatel nesmí do něj zasahovat v rozporu s prohlášením autora. Shareware je zpravidla mírně omezenou verzí plnohodnotného produktu (v tom se liší s demoverzemi, které jsou pouze ukázkou, v praxi nepoužitelnou), kterou lze po zaplacení registračního poplatku upgradovat na úroveň plnohodnotné verze (zpravidla zadáte heslo nebo číselný kód, načež zmizí omezení sharewarové verze). Výše registračního poplatku kolísá, ale většinou se pohybuje v rozmezí desítek USD (u menších programů typicky deset dolarů).

Freeware je také naprosto volně šiřitelný software, do nějž uživatel také nesmí dělat zásahy jdoucí proti autorovu prohlášení, ale oproti shareware není nijak omezen (je to plnohodnotný program) a, co je důležité, je úplně zadarmo (autor maximálně poprosí o ohlasy, komentáře a návrhy na zlepšení). Do této kategorie se počítá i Process Watcher.

Jako **public domain** se označují programy, které jsou volně šiřitelné, zadarmo, plně funkční, ale narozdíl od freeware do nich může kdokoli jakkoli zasahovat. Většinou jsou k volně dispozici zdrojové texty, které může si opět kdokoli upravit a dále šířit. Process Watcher není public domain, i když jeho zdrojové texty je možné za určitých podmínek získat.

Firmware je softwarové vybavení vestavěné do hardware, které zpravidla úzce s hardwarem souvisí. Je většinou silně specializované.

Halda a paměťový blok

Halda (heap) je část paměti určené pro alokaci paměťových bloků. Halda je spravována paměťovým manažerem (memory manager), který z ní přiděluje paměť po blocích potřebné velikosti. Proces může vlastnit několik hald, pak je jedna halda implicitní (default heap) a z ní paměťový manažer alokuje paměť, není-li dáno jinak. Halda může být i sdílená několika procesy (shared heap).
Paměťový blok (memory block, heap block) je část haldy určité velikosti, která má přiděleny určité příznaky, podle nichž paměťový manažer s tímto blokem zachází. Blok může být označen jako volný (je volně procesu k dispozici) nebo jako alokovaný (tj. aplikace jej používá a má v něm uložena svá data); alokovaný blok může být pohyblivý (movable) nebo nepohyblivý (fixed, unmovable, locked), nepohyblivý block je v paměti zamčen (locked) a paměťový manažer jej nemůže naprosto libovolně přemísťovat, pohyblivý blok může být přemísťován a odkládán na disk.

Handle

Přeložit anglický termín handle je velmi náročné. Přesný překlad je rukojeť, ale tento překlad se používá jen zřídka (kromě toho je dosti dlouhý). Handle je číselný odkaz na nějaký složitější objekt, pod nímž se můžeme na tento objekt odvolávat a pracovat s ním. Např. handle okna je jedinečné číslo okna, které se používá při všech operacích s okny. S handle se pojí většinou i určitá práva a omezení, která vyplývají ze způsobu získání handle. Handle je také omezený systémový zdroj.

Hint

Hints jsou malá okénka (zpravidla žluté barvy) sloužící jako stručná nápověda k ovládacím prvkům. Tyto popisky se objeví u myšího kurzoru, zastavíte-li kurzor na chvíli nad ovládacím prvkem. Program ovšem musí hints podporovat.

Hook

Hook je napojení Windows. Napojení Windows jsou řetězem procedur (hook procedures, hookproc), které jsou volány při aktivaci napojení. Napojení se aktivuje při nějaké události a dává možnost programátorovi ovlivňovat zprávy Windows ještě před doručením příslušné wndproc. Napojení mohou být i globální, pak zpracovávají zprávy ze všech procesů v systému. Mohou být i různých druhů (např. existuje napojení volané při pohybu myší, vytvoření a zničení okna atd.).

Módy procesoru

Procesory Intel kompatibilní mohou pracovat ve třech režimech, které se navzájem výrazně liší. Procesory Pentium (kompatibilní a vyšší jako Pentium Pro, Pentium II) pak mohou pracovat ještě v režimu čtvrtém.

Reálný mód (real mode) je jediný mód procesorů 8086, 8088, 80186, prvních intelovských procesorů, vyšší procesory jsou v tomto režimu degradovány na rychlé 8086. Je to režim, ve kterém běhá MS-DOS. Reálný režim se vyznačuje několika "přednostmi" a několika zápory.
Mezi klady (alespoň z hlediska programátora) je absolutní volnost – programátor se nemusí na nic ohlížet, paměť není chráněná proti zápisu, přístup na porty je naprosto volný. Jednoduchost a možnost aplikace pololegálních programovacích technik byla jako stvořená pro hry a pro viry (viry jsou samozřejmě méně populární než hry, ale virům to očividně nevadí a rádi se přiživí na popularitě i programových souborech her).
Nevýhody byly ale značné – paměť bez problémů dostupná jen do 1 MB, 64KB segmentace, faktická nemožnost multitaskingu (a multithreadingu ani nemluvě) nestandardizovaný přístup k zařízením, slabý výkon v některých oblastech, nedostatečná kvalita (a kvantita) služeb systému atd.

Chráněný mód (protected mode) je mód, v němž běží Windows (a další vyspělé operační systémy, jako Unix a jeho deriváty, OS/2 aj.). První procesor, který tímto režimem disponoval, byl procesor 80286 (ten ale měl chráněný režim díky 16bitové architektuře značně omezený); skutečný rozvoj chráněného režimu mohl nastat až příchodem procesoru 80386 (a Windows).
Nevýhodou je menší rychlost některých instrukcí (souvisí s ochranou paměti a nutností kontrolovat legálnost operací) a menší stabilita (to je ale mýtus zrozený díky Windows, jinak to není pravda) – možná by se ještě něco našlo.
Výhodou přístup teoreticky až k 64 TB (terrabytům!) paměti v 4 GB segmentech, teoreticky (a na některých systémech i prakticky) větší stabilita, podpora multitaskingu, podpora stránkování atd.

Virtuální 8086 (virtual 8086) je režim, fungující v rámci chráněného módu na virtuálním počítači. Umožňuje spouštění procesů reálného módu v chráněném režimu (např. spuštění DOSovského programu ve Windows).

Režim správy systému (system management mode, SMM) je jeden z režimů procesorů Pentium & spol., jenž umožňuje návrháři systému provádět určité vybrané důležité činnosti nezávisle na počítači a operačním systému; tento režim je určen pro firmware a normální programy jej nemohou využívat, už jen proto, že jej nelze zapnout softwarově, ale pouze při aktivaci hardwarově generovaného signálu SMI.

Mapping (mapování)

Mapování je vlastně zobrazení části paměti nebo souboru do adresového prostoru nějakého procesu, takže proces může s namapovaným objektem operovat, ač se fyzicky může nacházet jinde.
Namapovat lze např. modul (toho využívají hlavně Windows 95 a Windows 98), kdy kód modulu existuje v paměti jen jednou a je namapován všem procesům, které modul využívají, do jejich adresových prostorů; data má ale každý proces vlastní. Ve Windows NT jsou takto sdíleny jen systémové moduly, proto mají větší spotřebu paměti než Windows 95, ale jsou zase odolnější vůči chybám aplikací.
Další mapovatelný objekt je blok paměti, který může být díky mapování sdílen více procesy. Lze mapovat i soubor (resp. jeho část).

Modul

Modul je soubor s programovým kódem a/nebo daty v podobě zdrojů (resources), což mohou být obrázky, písma, texty apod. Každá Windows aplikace se skládá z několika modulů – z hlavního (spustitelný soubor, EXE) a knihoven (většinou DLL). Knihovny mohou být jak soukromé, které aplikace přinesla s sebou a jiné aplikace nevyužívají, tak i sdílené více aplikacemi (např. systémové knihovny využívají všechny aplikace).
Modulární struktura má své výhody i nevýhody. Výhodou je snadná rozšiřitelnost (a to i se zachováním zpětné kompatibility) a úspora paměti, protože kód (i data) může být sdílen a nemusí být duplicitní v různých aplikacích. Nevýhodou ale je ale občasná nekompatibilita některých verzí modulů způsobená nejčastěji chybou vývojářů, která někdy může značně zkomplikovat život. V modulární systém má také větší tendenci k chybám, které se hůře hledají.

Multitasking

Multitasking je metoda, jak na jednom počítači provozovat více programů současně, a to i v případě, že počítač má méně procesorů, než je spuštěných programů (úloh, tasks). Podstatou multitaskingu je to, že systém nechává běžet střídavě úlohy v krátkých časových intervalech, takže budí dojem, že běží současně.

První náznaky multitaskingu v systémech Microsoftu se objevují v MS-DOSu verze 2.0 – rezidentní programy. Rezidentní programy (tj. ty, které po ukončení zůstaly v paměti dále přítomny a "při životě" je udržovala různá přerušení, reakce na softwarové a hardwarové události) byly ale jen takovou berličkou, která budila zdání multitaskingu a časem vypučela v metlu uživatelů (každý neprimitivní virus pro MS-DOS je rezident).
Microsoft Windows 3.x (nižší verze v naších krajích snad ani nejsou známy a zapadly v zapomnění) byly už o krok dále. Umožňovaly tzv. kooperativní multitasking, už skutečný multitasking, kdy mohlo skutečně běžet víceméně plynule několik procesů. Procesy nebyly na sobě nezávislé, ale musely spolupracovat (proto kooperativní) – každý proces vykonal nějakou činnost a předal řízení dalšímu procesu v řetězu. V případě, že některá aplikace spolupracovat moc nechtěla, ostatní procesy jí byly brzděny; pokud dokonce některý program zamrzl, zpravidla nastal pád nebo zamrznutí Windows, protože spolupráce byla narušena a řízení nebylo předáno ze zamrzlé aplikace dalším procesům.
Windows NT a Windows 95 začaly konečně využívat preemptivní multitasking, multitasking, kde přidělování času aplikacím řídí jádro. Aplikace se nemusí starat o ostatní ani o to, zda má či nemá předat řízení dalšími procesu, jádro jí prostě po vyčerpání časového kreditu řízení odebere a přidělí ho jinému procesu. Společně s preemptivním multitaskingem se ve Windows objevil i multithreading.

Pro úplnost je nutno dodat, že jiné operační systémy (např. OS/2, Unix a systémy z něj odvozené jako Linux) disponovaly už dávno před Windows fungujícím preemptivním multitaskingem.

Multithreading

Multithreading úzce souvisí s pojmem thread. Jak název napovídá, jedná se o způsob, jak nechat běžet několik threadů současně, podobá se tedy velmi výrazně multitaskingu, nelze jej však s ním zcela zaměnit, protože thread není totožný s procesem a jeden proces může vlastnit více threadů.

Okno (window)

Okno je hlavním stavebním kamenem Windows. Oknem je ve Windows kde co, např. tlačítko Start je také oknem. Každé okno má několik atributů, které jsou nastaveny v závislosti na jeho specializaci; okna dokáží pracovat se zprávami a obsluhovat je pomocí wndproc.

Paměť

Paměť je dosti obecný pojem s řadou významů, které se navzájem prolínají. V širším slova smyslu je pamětí vše, co je schopné uchovávat informace (např. disk). V užším smyslu slova se pamětí myslí především paměť polovodičová umístěná na deskách uvnitř počítače – v tomto smyslu se také o paměti častěji hovoří. Rozdělit paměť můžeme z hlediska funkce a z fyzikálních (nebo chcete-li i fyzických) vlastností.

Základní dělení z pohledu hardwarového odborníka je na paměť ROM (read-only memory, paměť pouze pro čtení), jejíž obsah je stálý a do níž nelze zapisovat, nebo lze-li, pak pouze výjimečně a např. za použití speciálního zapisovacího zařízení, a paměť RAM (random access memory, paměť s náhodným přístupem), do níž zapisovat lze libovolně, ale jejíž obsah se po přerušení dodávky energie ztrácí. Pamětí ROM i RAM je spousta druhů např.: EEPROM, Flash EPROM,VRAM, DRAM, SRAM, SDRAM atd. Z hlediska využití lze paměť dělit na operační paměť, videopaměť, cache (na základní desce, procesoru, disku...) atd., podle toho, k čemu konkrétní čip slouží.

Nejdůležitější pro uživatele je operační paměť. V operační paměti je umístěn kód běžících programů a data, s nimiž programy právě operují. Ale i operační paměť lze různě dělit (a to je nyní to podstatné rozdělení).

Alokovaná paměť' je paměť přidělená z haldy paměťovým manažerem, kterou proces může využívat pro svá data. Poté, co ji přestane využívat, měl by ji uvolnit (dealokovat), aby ji mohly využívat i jiné procesy. **Volná paměť'** je pak paměť neobsazená (nealokovaná).

Pod pojmem **fyzická paměť'** myslíme paměť, která fyzicky existuje v počítači jako polovodičová operační paměť. Proč se zmiňujeme o fyzické paměti je patrné, až zmíníme-li pojem **virtuální paměť'**. Virtuální paměť je paměť teoreticky dostupná systému za použití odkládání na disk (swapování). Díky swapování lze používat více paměti, než kolik fyzické paměti skutečně v počítači máme. Cenou za použití většího množství paměti, než kolik jí ve skutečnosti máme, je značné zpomalení systému; operace s diskem trvají totiž řádově desítky milisekund (typicky 10 – 20 ms), operace s fyzickou pamětí desítky nanosekund.

Patch

Jako *patch*, tedy v překladu záplata nebo náplast (neplést s *path* – to je cesta), se označuje nějaké řešení, které dodatečně odstraňuje nějaké problémy, které se objevily po dokončení programu. Aby nebylo nutné vydávat znovu celý program, vydá se jen patch, který nalezené problémy řeší. Většinou je to nějaký menší program, který nahradí některé údaje programu či v něm něco přepíše, nebo soubor, jímž se nahradí jiný soubor. V případě Watchera je to soubor STATS.DAT, který napravuje chyby, na něž se dodatečně přišlo při testech na Windows 98.

Priorita

Každý proces a thread má určitou prioritu. Úroveň priority určuje, kolik času systém procesu nebo threadu přidělí a zda mu jej přidělí přednostně před ostatními procesy nebo thready. Čím vyšší priorita, tím více času proces/thread obdrží a tím větší přednost má před ostatními.

Procesy mají čtyři úrovně priority. **Nízká** (idle) priorita je určena pro procesy, které nejsou nijak důležité, běží v pozadí a většinu času čekají na akci uživatele (např. různé panely nástrojů apod. by měly využívat právě nízkou prioritu); tyto procesy běží jen tehdy, je-li systém v klidu, nebo jsou-li uspokojeny nároky procesů s vyšší prioritou. Nízkou prioritu má standardně Watcher při běhu v pozadí. **Normální** (normal) prioritu má většina aplikací. Je to obvyklá priorita zaručující poměrně hladký běh, není-li v systému nějaká překážka; má přednost před procesy s nízkou prioritou. **Vysokou** (high) prioritu má pak jádro. Proces s vysokou prioritou má zaručen běh, pokud je to alespoň trochu možné, a to i na úkor procesů s nižší prioritou (tj. normální a nízkou). Vysokou prioritu používá Watcher při své aktivaci, aby mohl zasáhnout i tehdy, bude-li zamrzlá aplikace blokovat systém. **Realtimová** (realtime) priorita je nejvyšší možnou úrovní priority. Proces s realtimovou prioritou má k dispozici tolik času, kolik je ho jen možné v rámci systému přidělit. Procesy s nízkou prioritou téměř neběží, procesy s normální prioritou jsou silně zpomaleny. Tuto úroveň priority používají hlavně hry a programy pro zpracování dat v reálném čase a např. rozhraní DirectX – tedy procesy, kde by jakékoli zpomalení mělo nežádoucí důsledky (komu by se líbila poškubávající se animace či přeskakující hudba).

Úroveň priority určuje podíl na procesorovém čase, který se každému procesu dostane – proces s vyšší prioritou poběží rychleji než proces s nižší prioritou. To ale neznamená, že proces s realtimovou prioritou má zaručen hladký běh! Pokud prostě počítač není dost výkonný pro danou aplikaci, pak ani realtimová priorita zbývající výkon nikde nevyčaruje.

Thready mají více úrovní priority než procesy. Priorita threadu je označuje jako tzv. delta priorita (delta priority), priorita procesu jako základní priorita (base priority). Výsledná priorita threadu se pak spočítá na základě obou těchto priorit a je určující pro thread v rámci celého systému. Bude-li mít thread nízkou základní prioritu (proces bude mít nízkou prioritu) a vysokou delta prioritu, bude na tom podobně jako jiný thread, jehož základní priorita bude vysoká a delta priorita nízká. Změnit prioritu threadu může ale v praxi jen vlastnící proces.
Thready mohou mít tyto úrovně delta priority: flákací (idle – omlouvám se za tento překlad, ale je to asi to nejvýstižnější a nejkratší, co mě napadlo), velmi nízkou (lowest), nízkou (lower), normální (normal), vysokou (high), velmi vysokou (highest) a maximální (time critical).

Proces

Jednoduše: proces je jiný název pro běžící program.
Složitě: proces je soubor threadů běžících ve společném adresovém prostoru, sdílejících tedy stejné moduly a paměť. Jeden z threadů je hlavním threadem procesu a každý proces musí mít alespoň jeden (hlavní) thread.

Registry

Registry (je těžké říci, jaký rod, jaké číslo přisoudit překladu anglického *Registry*, či jak vůbec rozumně toto slovo přeložit a nepoplést ho s jinými registry) jsou databází Windows, v níž jsou uloženy mnohé důležité informace jak o systému, tak o aplikacích. Registry mají (alespoň částečně) nahradit INI soubory, které byly v minulých verzích Windows značně rozšířené.
Narozdíl od INI souborů, je nelze editovat přímo, ale pouze pomocí programů pro jejich editaci (ve Windows 95 a 98 program REGEDIT.EXE, ve Windows NT REGEDIT32.EXE); pokud je chcete editovat, musíte vědět, co děláte, jinak můžete tak vážně narušit chod systému, že Vám třeba už ani nenastartuje. Mají stromovitou strukturu podobnou struktuře složek a souborů na disku. "Složky" se v Registrech nazývají *položky* nebo *klíče* a "soubory" *hodnoty*. Hodnoty mohou být textové (obsahují text), binární (obsahují binární data – tedy vlastně cokoli) a číselné (čtyřbytová celá čísla).
Fyzicky se Registry nacházejí v souborech USER.DAT a SYSTEM.DAT (popřípadě může být přípona i trochu odlišná např. DA0, v tomto případě se jedná o zálohu, nebo na víceuživatelských systémech to může být část Registru jiného uživatele), doporučeno je si tyto soubory velmi pečlivě chránit (a zálohovat).

Systémové zdroje

Za systémové zdroje považujeme především paměť, procesorový čas a zdroje systému (jako počet handle k oknům, souborů aj.).

Swapfile (odkládací soubor)

Odkládací soubor silně souvisí s virtuální pamětí. Aby Windows zabezpečily dostatek paměti a odlehčily paměti fyzické, odstraňují nepoužívané bloky paměti (tzv. stránky) z fyzické paměti a zapisují je na disk do odkládacího souboru. V případě, že odložená stránka potřeba, uvolní se pro ni dostatečný prostor ve fyzické paměti (např. odložením jiné stránky) a stránka ze načte z odkládacího souboru. Tímto je možno simulovat na počítači více paměti, než kolik má fyzické paměti. Popsaná operace se nazývá odkládání paměti (swapování).

Thread

Termín thread (vlákno) lze elegantně opsat jako podproces. Každý proces má několik (minimálně však jeden thread). Thread je samostatný proud instrukcí, nezávislý na ostatních threadech, se svým vlastním zásobníkem, frontou zpráv, ale sdílející s ostatními thready procesu stejný paměťový prostor, tedy i paměť, moduly a popřípadě další zdroje vázané na existenci procesu (mutexy, semafory, pipes, soubory atd.). Proces poskytuje tedy threadům rámec pro jejich fungování.

Tray icon

Tray ikona (tray icon – těžko přeložit, doslovný překlad, ikona na tácku, není asi nejlepší) je ikona umístěná vedle hodin v panelu úloh (zpravidla vpravo dole). Tray ikonu umísťují do panelu úloh programy, které běží v pozadí, chtějí, aby uživatel měl možnost je vyvolat, ale nechtějí zabírat zbytečně místo v panelu úloh jako tlačítko. Tray ikona může také poskytovat některé údaje (buďto už svým samotným obrázkem, nebo hintem, který se zobrazí po zastavení myšího kurzoru nad ikonou).

Virtuální počítač

Virtuální počítač je emulace dalšího počítače na reálném počítači. Spustíte-li ve Windows DOSovský program, je nutné napodobit prostředí DOSu do té míry, že se DOSovský proces musí cítit jako v reálném módu. Je vytvořen virtuální počítač, což je vlastně proces běžící v módu virtuální 8086. Na tomto virtuálním počítači (tedy v rámci procesu simulujícího reálný mód v prostředí módu chráněného) je teprve pak spuštěn DOSovský program. Windows aplikace běží na jediném virtuálním počítači, mohou tak běžet proto, že běží v chráněném módu ve víceúlohovém operačním systému.

Win16

Pod zkratkou Win16 můžeme chápat cokoli, co se váže k 16bitovým Windows (tj. Windows 3.x). Např. Win16 proces je proces určený původně pro 16bitové Windows.

Win32

Win32 je zkratka pro 32bitové Windows (Windows 95, 98 a NT). Win32 proces je pak proces určený pro 32bitové Windows.

Window message (zpráva okna)

Zprávy jsou jádrem funkcionality Windows. Každá zpráva detailně popisuje událost, která se v systému vyskytla. Po svém vytvoření a průchodem frontou systémových zpráv, kterou zpracovává systémový dispečer, je zpráva zaslána do fronty zpráv threadu, kterému je určena; každý thread má svou vlastní frontu zpráv (má-li vůbec frontu zpráv, mít ji nemusí), čímž se odstraňují problémy se společnou frontou zpráv z Win16. Ze své fronty zpráv thread odesílá zprávu svým oknům, pro než je určena, a ty ji dále zpracovávají pomocí své wndproc.
Příkladem zprávy může být pohnutí myší, stisk klávesnice, ale také zpráva o překreslení okna, jeho aktivaci či vytvoření, ukončení aplikace apod.

Window procedure (wndproc)

Wndproc (procedura okna) zpracovává zprávy, které přicházejí z fronty zpráv threadu, který okno vlastní. U zpráv, které vyžadují nějaké ošetření, provede wndproc určitou činnost, ostatní zprávy postupuje ke zpracování standardnímu mechanismu Windows.

Pro odborníky a kolegy takyprogramátory

Pokud patříte mezi odborníky či máte zkušenosti s něčím podobným, oč se Watcher snaží, ocením Vaše připomínky k mému řešení a návrhy na zlepšení – rád se něčemu přiučím.

Pokud patříte mezi zvědavce, experimentátory a programátory-amatéry, kteří by rádi zjistili více o tom, jak Watcher funguje a jak docílit toho či onoho (např. jak získávat výkonnostní statistiky), vězte, že Watcher byl psán coby seminární práce a plánuji jej zařadit do SOČ, je tedy věcí nikoli tajnou, nýbrž veřejnou a materiály uveřejněné v mé práci jsou dostupné. Zájemci o tyto materiály, kteří potřebují poradit, mohou počítat s mou pomocí.

Poznámka:
Zdrojové texty jsou napsány v Delphi 3.

Monitorování vzniku nových procesů

Zabezpečit tuto funkci pro mně znamenalo dosti experimentů a výsledek je v celku uspokojivý, ale myslím, že se to dá udělat i jinak a asi lépe. Jak zjišťuji vznik procesu já?
První možnost, jak tuto funkci zabezpečit byla triviální: nechám v pravidelných intervalech kontrolovat seznam běžících procesů a procesům, které mají běžet s určitou prioritou, tuto prioritu nastavím. Implementace jednoduchá, funkčnost zaručená, ale... Ale poměrně značné nevýhody. V první řadě zbytečně velké nároky na systém (kolik kontrolních cyklů by muselo proběhnout, než by se spustil hlídaný program, spustil-li by se vůbec, a kolik času by se tak zbytečně promrhalo zbytečnými kontrolami?). Druhá nevýhoda byla v tom, že uživatel mohl změnit ručně hlídanému procesu prioritu, a při další kontrole by se priorita vrátila na hlídanou hodnotu (a ošetření této chyby by bylo také dost náročné a ne příliš elegantní).
Začal jsem tedy přemýšlet, zda neexistuje nějaký mechanismus, který se aktivuje při spuštění procesu. Pravděpodobně něco takového existuje (protože existují programy, které také potřebují vědět, zda se nespouští nějaký jiný program a nepotřebují třeba ani žádný virtuální ovladač zařízení ale jen DLL). Věděl jsem, že existují napojení Windows (tzv. hooks), které umožňují získávat různé zprávy, které mohou dokonce patřit jinému procesu, než který vlastní napojení, a dokonce tyto zprávy měnit před doručením příslušné wndproc. Bohužel typy těchto napojení nebyly zrovna ideální pro mé účely, ale použít se daly – téměř každý program pro Windows vytvoří alespoň jedno okno (které nemůže patřit jinému oknu, protože je hlavním oknem tohoto programu) a to dokážu přece zjistit pomocí napojení! Nebylo ale zatím příliš důvodů k radosti. Od prvotní myšlenky byla dlouhá cesta k uspokojivému výsledku.
Po mnoha pokusech jsem dospěl k závěru, že optimální bude použít napojení typu WH_SHELL, kterým procházejí některé zprávy top-level oken, mimo jiné zprávy o vytvoření nebo zničení takového okna. Nepříjemné bylo zjištění, že kód pro obsluhu napojení musí být v DLL, aby bylo možné odchytávat zprávy z celého systému, ale ještě nepříjemnější bylo to, že takováto DLL mapovaná ke všem procesům, nemůže moc dobře používat nějaké proměnné, protože ani globální proměnné nejsou sdílené mezi procesy s namapovanou DLL. Musel jsem si osvojit file-mapping, aby si mohla DLL vůbec nějak uchovávat svá data, která musela být sdílená všemi jejími kopiemi. Bohužel ani toto řešení není ideální, ale ukázalo se jako nejlepší, které jsem dokázal vymyslet a zprovoznit – víte-li o lepším řešení, rád se poučím. Pro zájemce uvádím jádro knihovny MONHOOK.DLL:

```
// Knihovna MONHOOK.DLL - zkrácený okomentovaný text jádra knihovny
library MonHook;

...

resourcestring
```

```
 SMHDLLData = 'Process Watcher: MonHook DLL Data';
 SMHFatalError = 'Process Watcher: fatal error occured!';

...

// typ pro operaci s objektem file-mappingu
type
 PMHDLLData = ^TMHDLLData;       // ukazatel na TMHDLLData
 TMHDLLData = record
  MonHookHnd: HHOOK;             // uschované handle k napojení
  MappingObj: THandle;           // handle k file-mappingu, které jsme získali jeho vytvořením
  MainProcessID: Integer;        // ID procesu, který DLL inicializoval (Process Watcher) - jen
pojistka
  LastProcessCount: Integer;     // poslední napočítaný počet procesů
 end;

// hlavní funkce - hook callback
function MonHookProc(Code: Integer; wParam: WPARAM; lParam: LPARAM): LRESULT; stdcall;
var
 I, J: Integer;                // pomocné cyklové proměnné
 NewProcID: Integer;           // ID procesu okna, které vyvolalo zprávu
 MapHandle: THandle;           // handle k file-mappingu
 TempReg: TRegistry;           // pomocný objekt pro práci s Registry
 MappedData: PMHDLLData;       // ukazatel na namapovaná data
 MonitoredFiles: TStrings;     // pomocný seznam řetězců-jmen monitorovaných souborů
 TempList: TGSWeakObjectList;  // pomocný seznam pro uschování struktur s popisem běžících
procesů

begin
 // stejně musíme vrátit NULL
 Result := _NULL;
 try
  // otevřeme file-mapping
  MapHandle := OpenFileMapping(FILE_MAP_ALL_ACCESS, false, PChar(SMHDLLData));
  if MapHandle = _NULL then raise Exception.Create(_NULLStr);
  MappedData := nil;
  try
   // pokud jsme otevřeli úspěšně file-mapping, namapujeme si naše data
   MappedData := MapViewOfFile(MapHandle, FILE_MAP_ALL_ACCESS, 0, 0, 0);
   if not Assigned(MappedData) then raise Exception.Create(_NULLStr);
   // a s namapovanými daty můžeme pokračovat
   with MappedData^ do
    begin
     // pokud je okno vytvořeno, nebo se ničí, zajímá nás to
     if (Code = HSHELL_WINDOWCREATED) or (Code = HSHELL_WINDOWDESTROYED) then
      begin
       // inicializujeme si pomocný seznam
       TempList := TGSWeakObjectList.Create;
       try
        // a získáme do něj běžící procesy
        GetRunningProcesses(TempList);
        // pokud okno vzniklo, není to Watcher (který se právě aktivoval) a změnil se počet
procesů (to je částečná pojistka proti druhé nevýhodě
        // prvního možného řešení), pak prošetříme, zda se proces nenalézá mezi monitorovanými
        if (Code = HSHELL_WINDOWCREATED) and  (GetCurrentProcessID <> MainProcessID) and
(LastProcessCount <> TempList.Count) then
         begin
          // připravíme si pomocné objekty
          TempReg := nil;
          MonitoredFiles := nil;
          try
           TempReg := TRegistry.Create;
           MonitoredFiles := TStringList.Create;

           try
            // načteme jména hlídaných modulů
            with TempReg do
             begin
              OpenKey(SKeyWatcherMonitoring, false);
              GetValueNames(MonitoredFiles);
             end;
```

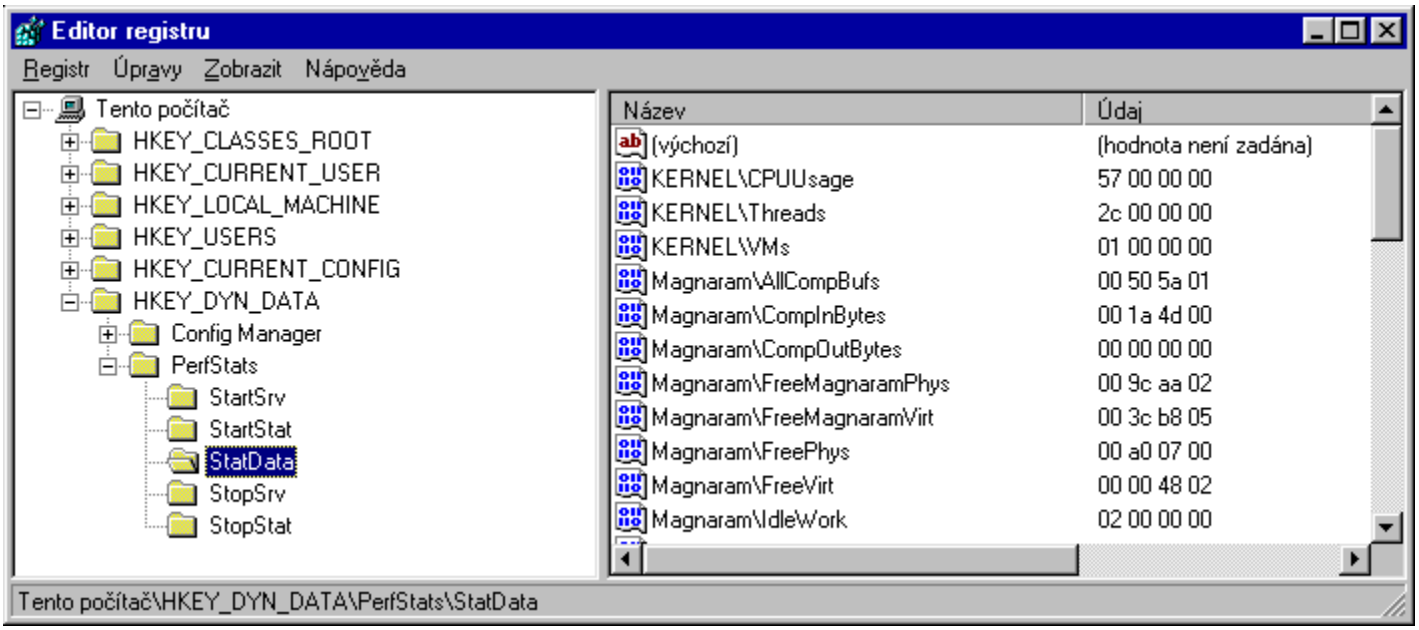
```
            // získáme ID procesu vzniknuvšího okna
            GetWindowThreadProcessID(wParam , @NewProcID);
            // a najdeme hlavní modul tohoto procesu
            with TempList do
             for I := 0 to Count - 1 do
              if TWeakProcessInfo(Items[I]).ProcessID = NewProcID then Break;

            // pokud byl nalezen, můžeme pokračovat, jinak máme smůlu
            if I < TempList.Count then
             // prohledáme seznam monitorovaných modulů
             with MonitoredFiles, TWeakProcessInfo(TempList.Items[I]) do
              for J := 0 to Count - 1 do
               // a v případě, že se jméno monitorovaného modulu shoduje s podezřelým modulem
               if CompareText(Strings[J], Filename) = 0 then
                begin
                 // nastavíme mu novou prioritu
                 SetProcessPriority(NewProcID,
TProcessPriority(TempReg.ReadInteger(Strings[J])));
                 Break;
                end;

           except
            // výjimky v hookproc raději ignorovat
            on ERegistryException do { nic };
           end;
          finally
           // uvolníme pomocné objekty
           MonitoredFiles.Free;
           TempReg.Free;
          end;
         end;

        // aktualizujeme počet procesů
        LastProcessCount := TempList.Count;
       finally
        // uvolníme další pomocné objekty
        TempList.Free;
       end;
      end;

     // zavoláme další hookproc v řetězu napojení
     Result := CallNextHookEx(MonHookHnd, Code, wParam, lParam);
    end;
  finally
   // a nakonec zrušíme mapování a file-mapping uzavřeme
   UnmapViewOfFile(MappedData);
   CloseHandle(MapHandle);
  end;
 except
  // ouha, výjimka - to je velmi zlé
  on Exception do
   // pokud není nalezen globální atom signalizující výjimku
   if GlobalFindAtom(PChar(SMHFatalError)) = _NULL then
    begin
     // tak si ho vytvoříme a zobrazíme upozornění uživateli
     GlobalAddAtom(PChar(SMHFatalError));
     MessageBox(_NULL, PChar(SFatalErrorMsg), PChar(SWatcherFatalError), MB_ICONERROR or MB_OK or
MB_TASKMODAL);
     // pak atom zase smažeme
     GlobalDeleteAtom(GlobalFindAtom(PChar(SMHFatalError)));
    end;
  // poznámka: globální atom musíme použít, protože kdyby, nedej bože, zobrazení upozornění
vyvolalo zase výjimku (a to je dost pravděpodobné), tak
  // se těch upozorňovacích dialogů na ploše naskládalo několik stovek a pak by nejspíš Windows
spadly, takhle si je uživatel bude moci resetovat
  // sám
 end;
end;

// funkce pro inicializaci monitoringu - vrací úspěch = true/neúspěch = false
```

```
function InitializeMonHook: Boolean; stdcall;
var
 // pomocné proměnné pro file-mapping
 MapHandle: THandle;
 MappedData: PMHDLLData;

begin
 // implicitně neúspěch
 Result := false;
 try
  // vytvoříme file-mapping
  MapHandle := CreateFileMapping(THandle($FFFFFFFF), nil, PAGE_READWRITE, 0, SizeOf(TMHDLLData),
PChar(SMHDLLData));
  if MapHandle = _NULL then raise Exception.Create(_NULLStr);
  MappedData := nil;
  try
   // namapujeme si data
   MappedData := MapViewOfFile(MapHandle, FILE_MAP_ALL_ACCESS, 0, 0, 0);
   if not Assigned(MappedData) then raise Exception.Create(_NULLStr);
   // a namapovaná data naplníme rozumnými hodnotami
   with MappedData^ do
    begin
     // handle použité k vytvoření file-mappingu
     MappingObj := MapHandle;
     // mateřský proces
     MainProcessID := GetCurrentProcessID;
     // handle k našemu napojení
     MonHookHnd := SetWindowsHookEx(WH_SHELL, TFNHookProc(@MonHookProc), HInstance, _NULL);
     if MonHookHnd = _NULL then raise Exception.Create(_NULLStr);
     // momentální počet procesů
     LastProcessCount := GetProcessCount;
    end;
   // pokud jsme to všechno zvládli, můžeme vrátit úspěch
   Result := true;
  finally
   // nakonec odmapujeme data
   if Assigned(MappedData) then
    if not UnmapViewOfFile(MappedData) then raise Exception.Create(_NULLStr);
  end;
 except
  // výjimky ignorujeme
  on E:Exception do { nic };
 end;
end;

// funkce pro ukončení monitoringu
function TerminateMonHook: Boolean; stdcall;
var
 // pomocné proměnné
 MappedData: PMHDLLData;
 TempHnd, SwapHnd: THandle;

begin
 // terminace bude implicitně úspěšná, protože co nám zbývá
 Result := true;
 try
  // otevřeme file-mapping
  TempHnd := OpenFileMapping(FILE_MAP_READ, false, PChar(SMHDLLData));
  if TempHnd = _NULL then raise Exception.Create(_NULLStr);
  MappedData := nil;
  SwapHnd := _NULL;

  try
   // namapujeme si data
   MappedData := MapViewOfFile(TempHnd, FILE_MAP_READ, 0, 0, 0);
   if not Assigned(MappedData) then raise Exception.Create(_NULLStr);

   with MappedData^ do
    begin
     // odložíme si původní handle, kterým jsem file-mapping vytvořili
     SwapHnd := MappingObj;
```

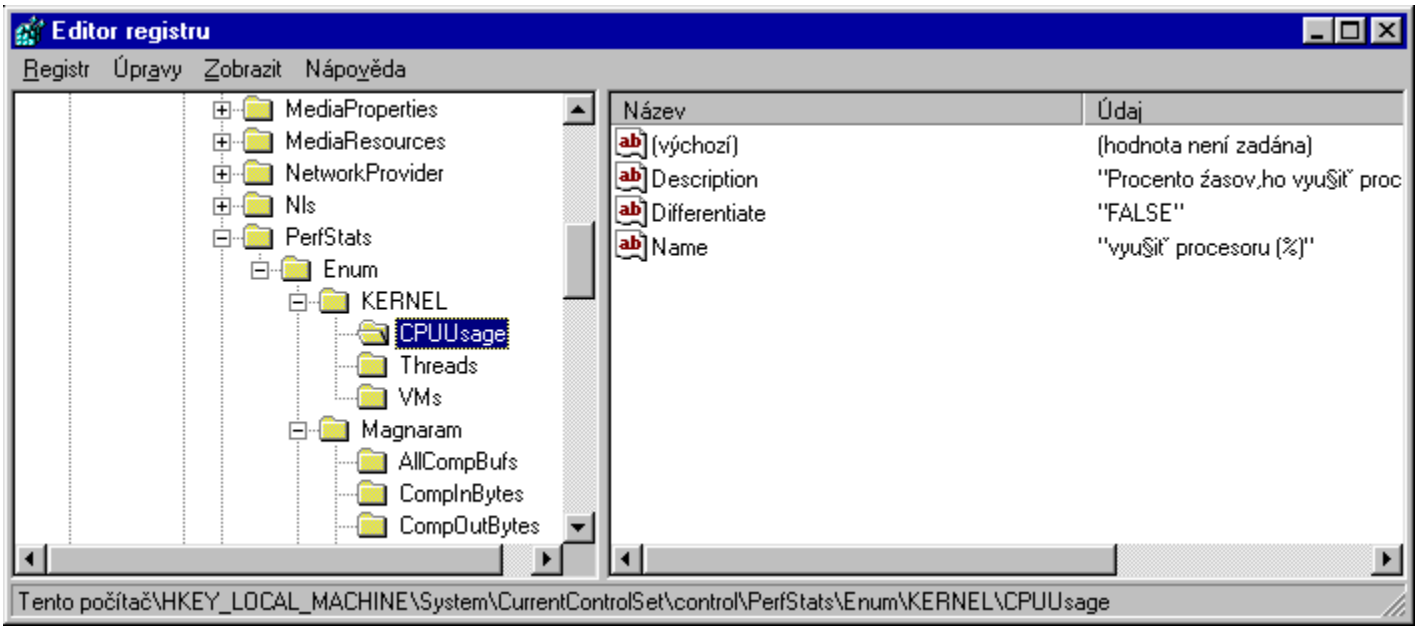
```
    // zrušíme napojení
    if not UnhookWindowsHookEx(MonHookHnd) then Result := false;
   end;
  finally
   // nakonec odmapujeme data
   if not UnmapViewOfFile(MappedData) then Result := false;
   // a uzavřeme všechna handle k file-mappingu, čímž ho zrušíme
   if not CloseHandle(TempHnd) then Result := false;
   if not CloseHandle(SwapHnd) then Result := false;
  end;
 except
  // případnou výjimku ignorujeme, jen vrátíme neúspěch
  on E:Exception do Result := false;
 end;
end;

// exportované funkce
exports
 InitializeMonHook,
 TerminateMonHook;

begin
 // tahle DLL žádnou zaváděcí inicializační rutinu nemá
end.
```

Problémy s přejmenovanými statistikami

Nejprve tedy snad k tomu, jak se statistiky ve Windows získávají. Ač je to nelogické, pomocí Registru. V klíči HKEY_DYN_DATA se ukládají dynamicky generovaná data, která se stále mění v závislosti na stavu systému (např. výkonnostní statistiky). Tento klíč má dva podklíče. K čemu slouží podklíč *Config Manager* jsem raději nijak nezkoumal; nás bude zajímat podklíč *PerfStats*. Ten má dalších pět podklíčů. Podklíče *StartSrv* a *StopSrv* jsem také dále nezkoumal (ale docela by mě zajímalo, k čemu jsou). Podstatné jsou zbývající podklíče *StartStat, StopStat* a *StatData*. Již názvy napovídají, k čemu jaký klíč slouží. Chceme-li získávat data nějaké statistiky, načteme z klíče *StartStat* příslušnou hodnotu (stačí jen několik bytů). Příslušná data pak získáme čtením stejnojmenné hodnoty v klíči *StatData*. Chceme-li skončit generování dat (což bychom nakonec měli, abychom systém zbytečně nezatěžovali), načteme opět danou hodnotu z klíče *StopStat*.



Čtyřbytové hodnoty z klíče *StatData* (data statistik) jsou sice typu binary data, ale po přetypování na celé čtyřbytové číslo, poskytují kýženou hodnotu. Statistiky jsou bohužel dvou typů: jeden druh má nastaven příznak *Differentiate* na hodnotu TRUE. U těchto statistik se hodnota v klíči *StatData* neustále zvyšuje (protože se neustále zvyšuje, je pravděpodobné, že jednou také přeteče a aplikace by měla s touto skutečností počítat) a vyjadřuje průběh v čase uplynulém mezi dvěma čteními (např. statistika VFAT\WritesSec, počet žádostí o zápis na disk: v určitých intervalech budeme načítat tuto hodnotu, počet žádostí o zápis v intervalu mezi oběma čteními je pak roven rozdílu načtených hodnot). Druhý druh (*Differentiate* je FALSE) poskytuje přímo aktuální hodnotu.
To, jaké je nastavení příznaku *Differentiate*, zjistíme v klíči HKEY_LOCAL_MACHINE\System\CurrentControlSet\control\PerfStats\Enum. Tento klíč má několik podklíčů, skupin statistik. Každá skupina statistik má pak podklíče, jednotlivé statistiky, v nichž jsou konkrétní údaje k dané statistice. Hodnota *Description* udává komentář ke statistice, *Name* stručné označení statistiky a *Differentiate* typ statistiky. Všechny hodnoty jsou textové. Jméno hodnoty statistiky v klíči HKEY_DYN_DATA... je pak rovno jménu skupiny + \ + jméno statistiky (např. KERNEL\CPUUsage – KERNEL je skupina, CPUUsage statistika).



Při testech na Windows 98 můj tester zjistil, že nefungují některé výkonnostní statistiky (konkrétně počet virtuálních počítačů a počet threadů). Po menším průzkumu jsem došel k závěru, že názvy statistik byly změněny (!) díky lokalizátorům (!!), kteří anglické názvy změnili na české (!!!), ale jen u dvou statistik a u ostatních nechali původní anglická jména (!!!!). Problém jsem vyřešil tak, že v Setupu lze po stisku tlačítka *Pomoc v nesnázích...* vyvolat dialog, ve kterém si uživatel může vybrat šablonu pro svůj operační systém, která přinutí Watchera používat nová jména statistik. Nevím, zda se podobný problém vyskytuje ve Windows NT nebo jiných lokalizacích Windows 98, pokud ano, sdělte mi to, prosím, společně s potřebnými údaji, abych mohl vytvořit novou šablonu pro Vaši verzi Windows. Zaslechl jsem (je to nepodložená spekulace), že snad existují překladové tabulky jmen statistik, ale nevím, kde jsou (jsou to funkce API, položky Registru nebo jsou někde úplně mimo?) a jak je používat. Kdybyste to někdo věděl, dejte mi laskavě vědět.

Potřeboval bych poradit s...

Rád bych do Watchera zabudoval v příští verzi několik nových funkcí, ale zatím nevím dost dobře jak na to. Víte-li to Vy, budu Vám vděčný, zašlete-li mi návod k řešení některého z uvedených problémů:

- Ü jak získat stav systémových zdrojů, prostředků GDI a USER (ale bez 16bitových funkcí, knihoven apod.)
- Ü podrobné informace o systému (procesor, sběrnice apod.) – předpokládám, že pomocí údajů v Registrech, ale co jsem zatím zjišťoval, nevypadá to vůbec příjemně
- Ü jak zjistit, zda je modul systémový nebo ne (viz Norton System Information)
- Ü monitoring by potřeboval vylepšit (pokud se tato funkce neujme a zaznamenám ohlasy, které neberou ani myšlenku, bude tento problém také vyřešen, protože pak jej z Watchera možná odstraním úplně)

Ze zákulisí

Z historie vývoje

Process Watcher je pokračováním programu Priority Watcher, který se zrodil na jaře roku 1998. Duchovním otcem Priority Watchera je Petr Adam, který měl sice mírně bláznivý, ale zajímavý nápad, jenž byl ale nad jeho síly, a tak jsem se jeho myšlenky na jeho žádost ujal. Priority Watcher byl jen takový pokusný projekt, který se ukázal být i životaschopný, nicméně příliš nedokonalý a nedotažený do konce. V létě 1998 o prázdninách popadla autora touha svůj prográmek vylepšit. Výsledkem byl Process Watcher ve verzi 0.9, který se nedopatřením dokonce dostal na internet (neboť testeři jej považovali za hotový, či téměř hotový). Ale verze 0.9 byla také jen ověřovací zkouškou (tímto žádám všechny, kteří si ji omylem stáhli z internetu, aby ji nahradili touto verzí, která už je opravdu definitivní; poznámka: nenechte se zmást úvodní obrazovkou, už tehdy jsem koketoval s myšlenkou, že někdy bude verze 1.0 a udělal úvodní obrazovku pro tuto verzi). Nakonec byla celá přepsána. Došlo k podstatným změnám designu a přidáním mnoha a mnoha zlepšení (automatické občerstvování bylo z hlediska programátora asi největší změnou, z hlediska uživatele je jistě více potěšující rozšíření výkonnostních statistik). Verze 1.0 by pravděpodobně vyšla značně později, ale zjistil jsem nemilou skutečnost, že jsem v posledním ročníku gymnázia, a že bych tedy měl napsat ročníkovou práci, popřípadě něco, co by se mohlo ucházet o místo v SOČ. Protože jsem se po dlouhých experimentech cítil být dosti obeznámen s problematikou procesů a částí WinAPI, která se jim věnuje, rozhodl jsem se, že budu pokračovat ve vývoji Process Watchera a dovedu verzi 0.9 zase o kousek dál.
Tato verze má číslo 1.01. Od verze 1.0 se liší vskutku jen nepatrně: byly opraveny (tentokrát snad už všechny) pravopisné chyby a jiné překlepy v nápovědě i v programu a především předělán Setup.

Co bude dál...

V plánu je mnoho dalších rozšíření, které z časových důvodů nemohly být do této verze zařazeny. Namátkou:

- Ü přidání PW (Process Watcher) skriptu (něco jako dávkový soubor pro Windows)
- Ü dodělání přídavných “rychlých” funkcí (aby oblíbené programy byly v menu, které se objeví po kliknutí na tray ikonu)
- Ü editor skriptů
- Ü shellová extenze umožňující spouštění procesů s danou prioritou za pomocí Průzkumníkova kontextového menu
- Ü plovoucí panel s výkonnostními statistikami (znáte Norton System Doctor?)
- Ü nový mechanismus získávání informací, aby byl méně náročný na paměť (na počítačích s menší pamětí mohlo snadno stát, že paměť došla)

A samozřejmě zohledníme návrhy a připomínky uživatelů. Pište a mailujte, určitě Vás vyslyšíme.

Lidé kolem Process Watchera

Petr Doležal
Hlavní (a jediný) programátor, autor nápovědy a všeho ostatního. Narozen 18. 5. 1981 v Třebíči, kde již sedmým rokem studuje víceleté gymnázium. Je už v posledním ročníku a čeká ho někdy v květnu 1999 maturita (všem, kteří mu budou držet pěsti pro štěstí, děkuje).

Petr Adam & František Folber
Testeři Process Watchera a autoři jedné velmi zajímavé stránky. Díky jejich laskavosti je i Watcher na internetu, právě na jejich stránce, kam Vás srdečně zvou (kontakty níže). Oba navštěvují třebíčské

gymnázium už šestým rokem (neznamená to ale, že by už aspoň dvakrát propadli – jen chodí do sedmiletého gymnázia).

Kontakty:

Petr Doležal
Vrchlického 647
Třebíč, Borovina
PSČ: 647 01
e-mail bohužel zatím není, elektronické příspěvky a komentáře adresované mně můžete zasílat na adresy testerů, kteří mi je předají

Petr Adam
e-mail: adamp@email.cz

František Folber
e-mail: isek@email.cz

Domovská stránka Process Watchera – Alias's (Shareware & Freeware) Archive (sekce Utility/Systémové nástroje)
http://alias.trb.czn.cz

Poděkování

Nejprve bych chtěl touto cestou poděkovat vedoucímu mé ročníkové práce, RNDr. Miroslavu Málkovi, za jeho laskavost, pochopení, připomínky a vůbec za vše, co pro mne kdy udělal. Díky, pane profesore.

Také chci poděkovat panu Vítu Kovalčíkovi (e-mail: vkovalcik@iname.com), autorovi několika utilitek pro Windows, za jeho nezištnou pomoc s problémem, se kterým jsem nemohl pohnout. Děkuji.

Pak bych chtěl pochopitelně poděkovat svým testerům a všem ostatním, které jsem zde nejmenoval (doufám, že se proto neurazí), ale jejichž pomoci si vážím.

A na závěr musím samozřejmě poděkovat uživatelům, kteří se rozhodli Process Watchera používat nebo alespoň vyzkoušet. Všechny Vaše komentáře a příspěvky, ohlasy negativní i pozitivní si pročtu a pokusím se na ně reagovat, případné dotazy se také pokusím v rámci svých skromných možností zodpovědět a odpověď zaslat co nejdříve. Díky.

Ještě omluva na závěr: vím, že má čeština silně utrpěla počítačovým žargonem; snažil jsem se psát tuto nápovědu co nejsrozumitelněji, a kde to bylo možné, držet se zavedené a známé terminologie nebo českých výrazů. I když se mi to asi nepovedlo tak, jak bych si přál, doufám, že se to dá číst a nápověda je skutečně nápovědou a ne něčím úplně opačným. Stejně tak se omlouvám za překlepy a další chyby, který jsem si nevšiml – snad jich není mnoho (a když tak to mohu svést na Murphyho zákony).

Bonbónek na závěr

Říká Vám něco věta *Jestliže se něco může nepodařit, tak se to také nepodaří*? Slyšeli jste ji už někdy předtím? Asi ano (a možná jste ji už prakticky ověřili), Murphyho zákony Vám tedy nebudou tak neznámé, možná ale neznáte Murphyho zákony aplikované na výpočetní techniku. Jestliže jste nenalezli odpověď na své problémy s počítačem v žádné odborné ani méně odborné publikaci, buďte si jisti, že je najdete právě tady.

Základy murphologie aplikované na výpočetní techniku

Murphyho zákon je upřesněním a logickým rozvinutím obecného zákona entropie, podle něhož se všechny částice universa snaží uspořádat se v co možná největším nepořádku. Poznání, že uvedené částice ti při tomto svém putování šlápnou alespoň jednou na palec, vedlo k formulaci

Murphyho zákona:
Jestliže se něco může nepodařit, tak se to také nepodaří.

Vynálezem počítače se člověk poprvé pokusil vdechnout neživé hmotě jistou inteligenci. Bylo to jistě fatální odhodlání. Neboť až dodnes nejsou počítače ani inteligentní ani kreativní. Avšak záludnost, lstivost a prohnanost jsou již u nich optimálně rozvinuty. Tak můžeme Murphyho zákon rozšířit o

První základní větu:
Murphyho zákon je počítačem optimalizován.

Protože však moderní počítače již dnes mohou provádět víc než jednu operaci současně, následuje bezprostředně

Druhá základní věta:
Všechno se nepodaří současně.

S vynálezem kontrolních součtů, korekčních programů, zálohovacích systémů a též systémů tolerujících chyby se před udiveným – na objekt degradovaným – člověkem otevírá mnohostrannost elektronického zpracování dat prostřednictvím

Třetí základní věty:
Nepodaří se to ani tehdy, když se to vlastně nepodařit nemůže.

Vyhodnotíme-li přebohaté zkušenosti uživatelů, programátorů, vývojových pracovníků a dalších ubožáků, můžeme nyní Murphyho zákon a jeho základní věty zákonitě aplikovat na běžný elektronický život:

První elektronická aplikace:
U počítačů není nic nemyslitelné, natož pak nemožné – kromě toho, co je potřeba.

Druhá elektronická aplikace:
Ve světě elektronického zpracování dat poruchy nekončí, nýbrž přecházejí jedna v druhou a navzájem se překrývají.

Třetí elektronická aplikace :
Komputerové poruchy čekají trpělivě na ten nejnepříhodnější okamžik, aby pak udeřily o to nemilostivěji.

Danielova korekční poznámka ke třetí elektronické aplikaci:
Může porucha udeřit?

Joachimova odpověď na Danielovu korekční poznámku:
Jen počkej.

Čtvrtá elektronická aplikace :
U počítačů se nedá na nic spolehnout.
Ani na to, že se nedá na nic spolehnout.

Pátá elektronická aplikace:
Nikdy se nemůžeš vyhnout velké poruše tím, že vyprodukuješ malou. V lepším případě se malá porucha připojí k velké, aby jí podpořila.

Šestá elektronická aplikace:
Nikdo si nedovede představit tolik poruch, kolik se jich odehraje uvnitř počítače.

Bernhardův povzdech:
Můžeš si však být jistý, že každou pocítíš jednotlivě na vlastním těle.

Bernhardův závěr:
Vskutku velké poruchy se chovají jako televizní intendanti:
1. Snaží se o co nejvíce opakování.
2. Levné poruchy neexistují.
3. Pokud by nějaká porucha byla levná, tak jsi dosud nepoznal její opravdový rozsah.

Jelikož počítače a elektronické zpracování dat měly alespoň původně cosi společného s matematikou, nebyl by tento obecný úvod do počítačové murphyologie úplný bez matematického zdůvodnění Murphyho zákona.
Čtenář nechť však vezme v úvahu, že matematika a binární a také lidská logika na jedné straně a zařízení pro elektronické zpracování dat na straně druhé, však také nemají nic, ale naprosto nic společného. Samozřejmě s výjimkou, že se pokoušíme dokázat následující větu:

Matematické zdůvodnění Murphyho zákona:
Exaktní matematický vzorec pro Murphyho zákon v oboru elektronického zpracování dat zní: 1 + 1 = 2
kde = je symbol s významem "zřídka, jestli vůbec kdy".

Teorém odchylky:
Rozdíl mezi digitální logikou a Murphyho zákonem spočívá v tom, že podle digitální logiky by se vlastně muselo vycházet z toho, že všechno se nedaří vždy podle stejné metody.

Binární překlad teorému odchylky:
Jestliže je nula obzvlášť velká, pak je téměř tak velká jako kousek jedničky.

Pachatelé

Když se kdekoliv na světě setkají počítač a člověk, jsou tam také pachatelé a oběti. Kdo je pachatel a kdo oběť, určuje osobní vztah: Je úplně jedno, co se stane, ty budeš vždy patřit k obětem. Jako všude v životě platí to konec konců i ve světě počítačů.

Obecný zákon oběti:
Lhostejno, na které straně právě stojíš – tato strana prohraje. Jestliže přejdeš na druhou stranu, otočí se i válečnické štěstí.

Obecný zákon oběti, precizovaný a aplikovaný na obor elektronického zpracování dat, dává
Pravidlo kvartetu:
1. Jsi-li uživatel, prohraješ s počítačem, výrobcem hardware i s programátorem.
2. Jsi-li výrobce hardware, prohraješ s počítačem, uživatelem a programátorem.

3. Jsi-li programátor, prohraješ s počítačem výrobcem hardware a uživatelem.

Důsledné závěry z kvartetového pravidla:
1. Nemohou existovat vítězové – lidé.
2. Počítač zvítězí vždy.

Rozšířený závěr z kvartetového pravidla:
Pokud by někdy počítač nezvítězil, pak vyhraje software nebo periferní zařízení, v nejlepším případě elektrická zásuvka.

1.0 Hardware

Hardware je vydařený pokus, jak předem vytušit softwarové chyby, optimalizovat již existující omyly, vše uložit do paměti a produkovat stále větší rychlostí. Hardware se skládá z počítače, vstupního zařízení, tiskárny, paměti a dalších do křemíku zafixovaných záludností. Z pohledu uživatele je zde hardware proto, aby analogicky zadané úloze spolehlivě a co nejvyšší rychlostí vytvořil tolik chyb, že v nejkratší možné době dojde k nejvyššímu možnému počtu již nenapravitelných škod.
Naopak pro programátory a výrobce je hardware proto, aby analogicky zadané úloze spolehlivě a co nejvyšší rychlostí vytvořil tolik chyb, že v nejkratší možné době dojde k nejvyššímu možnému počtu již nenapravitelných škod.

Dvojzákon komplexního hardware:
1. Komplexní systémy inklinují ke komplexním chybám.
2. Jednoduché systémy naopak inklinují ke komplexním chybám.

První doplnění:
Nové systémy produkují nové chyby.

Druhé doplnění:
Nové systémy opakují svoje nové chyby.

Třetí doplnění:
Staré systémy produkují nové a staré chyby.

Závěry:
1. Komplexní systémy inklinují k tomu, aby omezovaly svou vlastní funkci.
2. Počítače fungují jen proto, aby mohly produkovat chyby.
3. Systémy inklinují k růstu, a stávají se tak troufalými.

Speciální upřesnění pro oddělení:
Chceš-li mít na svém oddělení permanentní výmluvu pro vlastní chyby, vybav je počítačem (viz též odstavec *Nejlepší výmluvy*).

1.1 Počítač

Výkonnost počítače je dána jeho

inteligencí	tedy počtem pevně zabudovaných chyb,
rychlostí	tedy počtem katastrof, které je schopen vyprodukovat za jednotku času,
dobou odezvy	tedy dobou, kterou počítač potřebuje, aby se zotavil z tvého vstupu.

Už léta se počítačový průmysl – a to úspěšně – snaží zvyšovat výkonnost a náchylnost k chybám u svých systémů a tím poskytnout obživu stále většímu počtu operátorů, servisních podniků, montérů a expertů na problémy. Proti tomuto rostoucímu trhu stojí na druhé straně stále se zmenšující počet firem, které jsou ještě schopny i bez počítačů

dosahovat pozitivní bilance. Stranou ponechme ještě menší množství firem, které jsou s to i navzdory svým počítačům dosahovat pozitivní bilance.

Murphyho zákon (že totiž všechno, co se může nepodařit, se také nepodaří) je, jak jsme konstatovali, optimalizován počítačem. Protože toto pravidlo platí jak pro samotný počítačový systém tak i pro vztah "počítač – zbytek světa", existuje obrovská naděje, že již v dohledné době nebudou kromě výrobců počítačů a servisních podniků existovat žádné jiné firmy. A až i prvně uvedené ohlásí na základě obligatorních počítačových chyb konkurs, zůstane jen společnost, ve které nebude existovat nic, než počítačové chyby.

1.2 Vstupní zařízení

Počítačový průmysl s oblibou označuje klávesnici, myš, trackball a tablet za slabé místo uživatele. Při důsledné aplikaci Murphyho počítačových zákonů to vlastně nemůže znamenat nic jiného, než že se uživatel "sekl", když se domnívá, že by mohl s některým z těchto zařízení rozumně pracovat.
Zatímco klávesnice byla stvořena k tomu, aby pomocí nelogických pohybů kurzoru volila z nepřehledných menu sotva srozumitelné příkazy, které by bylo jednodušší navolit myší, slouží myš, trackball a tablet k tomu, aby pomocí nelogických pohybů byly z nepřehledných menu vybírány těžko srozumitelné příkazy, které je snazší navolit pomocí klávesnice, a přitom se vytvářely tak dlouhé a komplikované větné konstrukce, až se lektorovi, čtenáři a textovému editoru udělá špatně.
Majitelé počítačů, kteří skutečně vidí do budoucna, proto čekají na první funkční zařízení s hlasovým vstupem. Teprve pomocí těchto aparátů bude možné dosáhnout nejvyššího stupně neporozumění mezi počítačem a jeho obsluhou.

Oběti

Počítačový průmysl je společnost spiklenců. Programátoři, fanatici a výrobci hard- a software se spikli proti uživatelům. Uživatelé, fanatici a programátoři se spikli proti výrobcům. Fanatici, uživatelé a výrobci se spikli proti programátorům. A všichni společně se spikli proti lidem, kteří o počítači vědí jen to, že zabírá místo na psacím stole.
V tomto věčném boji přirozeně nemůže nikdo zvítězit. Lépe řečeno: nemůže nikdy zvítězit člověk. Neboť přes všechny záludnosti, zafixované v bitech a vodičích, zůstane počítač jako smějící se třetí, zcela nezasažen.
Principiálním poznatkem všech postižených elektronickým zpracováním dat – zkrátka všech obětí – je:

Základní poznatek o využití elektronického zpracování dat:
Počítač existuje proto, aby ti usnadnil práci, která by bez něho neexistovala.

1. Programátoři

Programátoři jsou (i přes protichůdné pověsti) lidé, kteří se v pozdních nočních hodinách, určených ke spánku, pokoušejí pomocí zcela nevhodných vývojových balíků pro neslučitelné slepence chybami zamořeného hardware přeměnit navzájem si odporující požadavky uživatelů v programy, které nakonec nikdo nepoužije.
Programátoři se dělí do dvou kategorií. První skupina se pokouší za příliš málo peněz a s příliš velkou námahou postavit proti sobě logické omyly programovacích jazyků, chyby kompilátorů a do křemíku zatavené nesmysly hardwarových vývojářů tak, až počítač nakonec alespoň občas dělá to, co se od něj očekává. Druhá skupina dělá totéž, avšak bezplatně.
Obecně je programátorovo myšlení logické ("if 1 = 2 call Mainprogram"), vždy strukturované ("on Hlad gosub Lednice else return") a nezatížené předsudky. Proslýchá se, že se prý snad vyskytují jednotliví zástupci této profese, kteří se nezbavili předsudku, že počítač byl stvořen k tomu, aby sloužil člověku. Nebo, jak to vyjádřil slavný angloamerický spisovatel William D. Base Shakespeare: 2 B or not 2 B.

I když je obtížné interpretovat Murphyho počítačový zákon z hlediska programátora (konec konců programátor je v podstatě zcela zbytečným článkem řetězu – marketing – reklama – programátor – odbyt – uživatel – podpora – update), pokusme se o to na následujících stránkách. I přesto, že se softwarové firmy a uživatelé již dávno shodli na

tom, že jejich život by byl mnohem snazší bez příliš dobře placených programátorů a jejich věčných námitek stran proveditelnosti jistých nároků na programy. Renomovaní výrobci proto už dávno s úspěchem přešli k tomu, že zadávají vývoj software přímo prostřednictvím *CASE* ("Computer Aided Software Engeneering"), poněvadž konec konců jen počítač dokáže psát programy tak, že mu jiné počítače mohou správným způsobem neporozumět.

Lückeova podstata programování:
Nebude to fungovat.

První upřesnění:
Pokud to přece jen funguje, naprogramoval to někdo jiný.

Druhé upřesnění:
Jedinými jazyky, které všichni programátoři ovládají perfektně, jsou klení a nadávání.

Závěr:
Počítač udělá to, co naprogramuješ, ne to, co chceš.

Zdvojené pravidlo pro programátory – amatéry:
1. Předvádíš-li vlastnoručně sestavený program, pak při prvním předvedení narazíš na podstatnou chybu.
2. Největší chyby nedokážeš odhalit. Zato si jich ihned všimne každý, kdo poprvé spustí tvůj program.

Axelův poznatek o ladění:
Nic nevylepší žádný program víc, než absence kontrolních rutin.

Axelův doplněný poznatek:
Je-li ladění metoda k odstraňování chyb z programu, pak je programování metoda, jak chyby do programu zabudovat.

Axelův závěr:
Když nevíš, co děláš, dělej to s noblesou.

Petrův zákon o "špagetovém" kódu:
Programový propletenec narůstá do té míry, až překročí schopnosti programátora, který jej vyvíjí.

Dodatek k Petrovu zákonu o špagetovém kódu:
Přípravné práce prováděly vždy osoby, které jsou na nejlepší cestě dosáhnout nejvyššího stupně vlastní neschopnosti.

Syndrom kvality:
Každý program, který dobře začíná, špatně skončí.
Projekt, jehož programování začíná špatně, skončí katastrofálně.

Závěr 1:
Co vypadá jednoduše, je složité.
Co vypadá složitě, je nemožné.
Co vypadá nemožně, dokáže vyřešit i uklízečka, a to bez počítače.

Závěr 2:
Nikdy není tak zle, aby nemohlo být ještě hůř.

Závěr 3:
Uklízečka začínala kdysi dávno u konkurence jako systémový programátor.

Zákon procedur:
1. Každá programová procedura, do níž se může vloudit chyba, bude samozřejmě chybu mít.
2. Též v procedurách, které musí být bez chyby, jsou chyby.

První závěr:
Každá chyba bude skryta tam, kde bude objevena co možná nejpozději a kde způsobí co možná největší škodu.

Druhý závěr:
Každá chyba se projeví teprve tehdy, až celý program projde poslední kontrolou.

Třetí závěr:
Bude-li chyba odhalena dřív, není možno najít příčinu.

Zákon konce:
Dokončení programu vyžaduje vždy dvojnásobek předpokládaného času. Respektuješ-li tento zákon při sestavování časového rozvrhu, pak platí věta o rekurzi.

Zákon o změně programu:
Čím jednodušší se zdá změna být, tím větší okruhy zasahuje a tím více procedur musí být napsáno znovu.

Zpřesnění zákona o změně programu:
1. Když se chce, tak to nejde.
2. Nic není tak jednoduché, aby se to nedalo pokazit.

Pravidlo zachycení:
Jestliže vyvineš proceduru, která vychytá zjevné chyby před výstupem, najdou se uživatelé, kteří ji obejdou tak, že si chybné výstupy obstarají předem.

Zákon o vynalézavosti uživatele:
Je-li zjištěno, že existují 4 různé možnosti, jak přivést program ke zhroucení, a jsou-li všechny čtyři odstraněny, najde první uživatel pátou možnost.

Zobecnění:
Každý program můžeš zabezpečit před blázny, ale žádný nedokážeš zajistit stoprocentně.

Zákon o dokumentaci:
Návody nikdo nečte.

Výjimky:
1. Špatné návody čtou recenzenti.
2. V návodech se obvykle čtou jen ty pasáže, které přimějí uživatele dělat chyby.
3. Každý návod zastará v okamžiku vydání.

Axiom vztahu návod/program:
I když něco vysvětlíš tak přesně, že tomu nikdo nemůže neporozumět, přesto tomu někdo rozumět nebude.

Vogelova souvislost mezi recenzí a návodem:
1. Sestavíš-li dobrý návod, nepovažuje jej recenzent za dostatečně podrobný.
2. Sestavíš-li dobrý a podrobný návod, nebudou návody v testech hodnoceny.
3. Sestavíš-li špatný návod, bude to rozhodujícím kritériem pro recenze ve všech odborných časopisech.

Dogma o zákeřném algoritmu:
Jestliže program funguje, pokazilo se něco už předem.

Závěry z dogmatu o zákeřném algoritmu:
a) Je jedno, co se pokazí, v každém případě to bude vypadat správně.
b) Ten, koho poprosíš o pomoc, chybu nenajde.
c) Náhodně příchozí objeví chybu hned, aniž bys ho o radu žádal.
d) Je jedno, co se nepodaří; vždy je tady někdo, kdo to věděl předem.
e) Nevěř na zázraky – spolehni se na ně.

Poznatek aplikačního programátora:
Zásada: uživatel dělá všechno špatně.
1. Když napíšeš "Napiš < J > nebo < N >", napíše "< J > nebo < N >".
2. Když napíšeš "Stiskni < Return >", napíše "< Return >".
3. Když napíšeš "Stiskni libovolnou klávesu", zmáčkne Shift nebo NumLock.

2. Uživatel

Obvykle je uživatel definován jako periferní zařízení, jež se snaží pomoci nedostatečného hardware a nesrozumitelného programu vyřešit problém, který by se bez počítače dal vyřešit dvakrát rychleji. Takovéto vymezení pojmu je však nanejvýš nepřesné a povrchní. Ve skutečnosti je uživatel periferní zařízení, jež se snaží pomocí nedostatečného hardware a nesrozumitelného programu vyřešit problém, který by bez počítače vůbec neexistoval.

Tento úkol uživateli usnadňují:
a) české národní prostředí
b) uživatelsky příjemný optimální hardware a software
c) množství obsáhlé, leč vzájemně si odporující odborné literatury.

České uživatelské prostředí a česká verze programu se zpravidla skládá ze špatně přeloženého návodu, zkomolených systémových hlášení a z anglických příkazů.
Pod uživatelskou příjemností je třeba rozumět zdvořilé, ochotné a trpělivé chování uživatele k nafoukanému, záhadnému a nepřizpůsobivému hard- a software. Úkolem uživatele však není hledat cesty k řešení, nýbrž v prvé řadě přijít na to, proč program a hardware dělají něco jiného, než je psáno v příslušné dokumentaci.
Navzdory obecné platnosti Murphyho počítačových zákonů není pro uživatele v zásadě nemožné rozmotat spletité vztahy hard- a software. Že se to v dějinách počítačů dosud nikomu nepodařilo, není však definitivním vyvrácením této věty.
I když je obtížné interpretovat Murphyho počítačový zákon z hlediska uživatele (koneckonců uživatel je v podstatě zcela zbytečným článkem řetězu vývoj – programátor – výrobce – uživatel – servis – výkupna počítačového šrotu), pokusíme se o to v následujících stránkách. A to přesto, že se pracovníci vývoje, programátoři a výrobci již před mnoha lety shodli na tom, že bez uživatelů by byl jejich život mnohem snazší.
Renomovaní výrobci s úspěchem již dávno vyvíjejí takový hard- a software, který nebere ohledy na zmíněný, chybami prolezlý článek řetězu.

První zásada použití počítače:
Když se něco nedaří, víš jen, že ses dopustil lichého počtu chyb.

Druhá zásada použití počítače:
Počet chyb n v libovolném počítači, respektive (libovolném programovém balíku) lze přesně vypočítat podle vzorce: $n > a$, kde a je libovolně zvolené číslo.

Třetí zásada použití počítače:
Jestliže se něco daří, pak je počet chyb větší než $n+1$.

Čtvrtá zásada použití počítače:
Když už nefunguje vůbec nic, přečti si konečně návod k použití.

Axiom o rozmnožování problémů:
V každém velkém problému je skryt jeden malý, který by rád ven.

Shainkerův opak:
V každém malém problému je skryt jeden velký, který by rád ven.

Joachimův hluboký povzdech:
I tam, kde není vůbec žádný problém, je skryt jeden velký, který by rád ven.

Shrnutí

Vážený počítačový příteli!
Aplikovaná filozofie katastrof přináší nový pohled: Murphyho zákony jsou optimalizovány počítačem.Veškeré myslitelné složky, které mohou způsobit poruchu jsou v počítači soustředěny: procesor, monitor, paměť, přídavné karty, klávesnice a další periferie se společně snaží o to, aby byly co nejméně kompatibilní. A kde toto vše ještě nestačí, tam je triumvirátem programátor – uživatelský program – uživatel podporován fyzikální zákon entropie, podle něhož příroda vždy usiluje o stav maximálně možného chaosu.

Závěrečný poznatek:
V boji mezi tebou a digitálním světem se přidej raději na stranu digitálního světa.

Poznámka:
Převzato z knížky Joachima Grafa *Murphyho počítačové zákony aneb jak počítač optimalizuje zákon o tom, že se pokazí vše, co se pokazit může* (1. vydání v češtině: Unis Edition, 1992)

Ruční odinstalace

V případě, že Setup selže, je dobré odstranit zbytečné položky v Registrech ručně. Uživatelům Windows NT doporučujeme po odinstalaci Watchera zkontrolovat, zda jim v Registrech nezbyly zbytečné položky, které Setup neodstranil.
Odinstalaci provedete tak, že spustíte program REGEDIT.EXE (Windows 9x) nebo REGEDIT32.EXE (Windows NT), popřípadě jiný ekvivalentní program (např. Norton Registry Editor). V levé části je stromovitá struktura tzv. klíčů, vpravo pak k otevřenému klíči tzv. hodnoty, které otevřený klíč obsahuje. Je to podobné uspořádání souborů a složek na disku – jako složky figurují klíče, jako soubory hodnoty. Watcher vytváří tyto klíče:
HKEY_USERS\xxx\Software\G-SOFTware
HKEY_USERS\xxx\Software\G-SOFTware\Process Watcher 1.0
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\Process Watcher
(xxx v klíči HKEY_USERS je jméno konkrétního uživatele)

a hodnoty:
klíč: HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
hodnota: Process Watcher

Klíč *HKEY_USERS\Software\G-SOFTware* smažte jen tehdy, není-li v něm jiný další podklíč než *Process Watcher 1.0*. V podklíčích klíče *HKEY_USERS* se skladují uživatelská nastavení, dojde-li k opakovaným problémům s nastaveními, zkuste klíč *Process Watcher 1.0* v příslušné větvi klíče *HKEY_USERS* smazat – sice ztratíte svá nastavení, ale pravděpodobně i problémy.
Smazáním výše uvedených klíčů (s přihlédnutím k předchozímu odstavci) zlikvidujete veškeré stopy Watchera v systému (jiné zásahy Watcher opravdu nedělá). Může se ještě maximálně stát, že zůstane někde neplatný zástupce (ve Windows 9x nejspíše proto, že jej někdo umístil jinam, než kde jej hledal Setup, ve Windows NT pravděpodobně pro nedostatečnou úroveň oprávnění k smazání příslušného zástupce nebo nemožnosti přístupu k němu).