Pohled do jádra: Jak silná jsou Windows

Kdo chce opravdu vědět, jak dobrý je jeho operační systém, musí se podívat do jádra. A také musí rozumět informacím, které zde najde. Srovnali jsme **SILNÉ A SLABÉ STRÁNKY** operačních systémů Windows, Linux a Mac OS X.

JÖRG GEIGER

Windows – toto slovo v mnoha uživatelích vyvolává vlnu nevole. Je s ním však spojena i řada mýtů, předsudků a polopravd. Na internetu jsou miliony virů, které útočí jen na operační systém od Microsoftu. Navíc všichni vědí, že čím déle operační systém běží, tím hůř se chová. A co se stability týče, je označení „modrá obrazovka" synonymem pro Windows. Není divu, že se tak děje, vždyť Vista obsahuje 70 milionů řádků kódu. V takovém kódu se pak nemůže vůbec nikdo vyznat.

Jsou tyto názory pravdivé, nebo je to jen do očí bijící hloupost? Abychom si situaci objasnili, budeme se muset podívat do jádra systému a zhodnotit tři základní kritéria: bezpečnost, výkon a stabilitu. A neprovedeme to jen u Windows, ale také u dvou konkurenčních platforem – u Linuxu a Mac OS X. Uvidíme, který operační systém je na tom nejlépe.

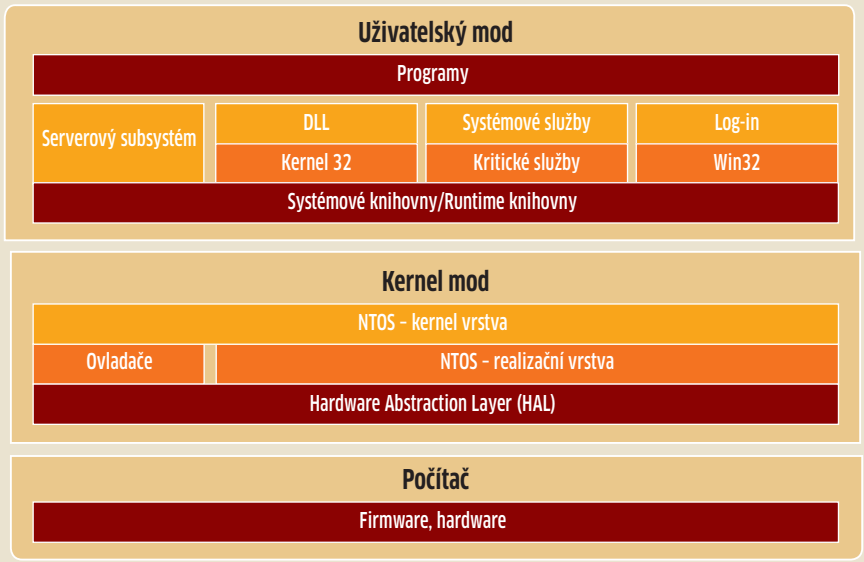
Jádro: Vládce systému

Celý operační systém řídí kernel neboli jádro systému. Jeho kvalita pak rozhoduje o kvalitě celého systému. Udržuje systém v běhu a zajišťuje přístup k hardwaru. Přesněji: jádro komunikuje s externími zařízeními, spravuje vnitřní komponenty, jako je RAM, CPU a pevný disk, a také řídí veškeré procesy a jejich požadavky na zdroje.

Má-li být systém bezpečný, musí kernel pečlivě kontrolovat všechny procesy, které spouštíte a které běží. Kernel totiž určuje, které programy a po jakou dobu budou mít přístup k hardwaru. Aby byl systém stabilní, používá se strukturování zdrojů. A vše musí probíhat v reálném čase, tak jak pracujeme každý den na počítači. Například když se dva programy pokouší zapsat informace na pevný disk, jádro musí rozhodnout, který program zapíše svoje informace první a který program bude muset chvíli počkat. K rozhodnutí, které musí být učiněno za nesku-

Architektura Windows

Mnoho oblastí operačního systému od Microsoftu je pro uživatele nedostupných. Při své práci může vidět jen to, co běží v uživatelském režimu. Co se odehrává v jádře, to je pro uživatele neviditelné.



70 milionů řádků kódu – to je Vista

tečně krátký okamžik, přitom vede několik posouzení ze strany jádra. My se podrobně podíváme, jak to funguje ve Windows.

Windows: Běží na každém hardwaru

Architektura Windows se od zavedení NT dělí na dvě vrstvy – uživatelský mod a kernel mod. To samozřejmě platí i pro Vistu. V uživatelském modu pak běží vše, co je viditelné pro uživatele. Jsou to tedy programy, jako je třeba Word nebo Photoshop. Tyto programy nemají žádný přímý přístup k hardwaru ani k operační paměti. Aplikace v uživatelském modu jsou proto jako v bavlnce. Veškeré požadavky, které procesy mají, jsou postupovány hlouběji do systému a starají se o ně příslušné části systému. Tím může být třeba Win32-API se svými DLL knihovnami.

Kernel mod naproti tomu běží na pozadí, což uživatel sedící u počítače vůbec nevidí. Tedy až do okamžiku, než třeba ovladač grafické karty shodí celý systém a uživatel uvidí modrou obrazovku.

Centrem celého kernelu je soubor ntoskrnl.exe. Ten od sebe odděluje kernel mod a uživatelský mod. Jedním z úkolů je přidělování času CPU mezi oba mody. Nějakou část výkonu totiž spotřebují procesy v kernel modu a nějaký čas potřebují běžící

uživatelské aplikace. Kromě toho obstarává ještě celou řadu dalších systémových služeb, jako je třeba plug & play.

V nejhlubší části systému se pak nachází vrstva Hardware Abstraction Layer (HAL). Ta vlastně vytvoří vrstvu těsně nad hardwarem, díky které se velmi zjednoduší psaní ovladačů i aplikací. Je totiž jedno, jestli je hardware tvořen starým AMD procesorem, nebo novým čtyřjádrovým Intelem. Windows zde poběží. Pokud by vrstva HAL v systému neexistovala, musel by Microsoft vydávat speciální distribuci Windows pro každý počítač na světě!

Linux: Moduly na přání

Linuxové jádro je založené na Unixu. Jádru z Windows je ale více podobné, než si možná myslíte. Linux je totiž také usazený přímo na hardwaru a funguje jako vrstva mezi hardwarem a programy uživatelů. Zde bychom rozdíl mezi Linuxem a Windows hledali jen těžko. Stejně jako jádro Windows totiž používá vstupně-výstupní (V/V) zařízení dohromady s pamětí.

V Linuxu se vše točí kolem souborů, procesů a jejich přerušení. Všechno musí být vyvážené. K přerušení dojde třeba v okamžiku, kdy uživatel stiskne nějakou klávesu na klávesnici. Tím se vyvolá přerušení a speciální mechanismus musí rozhodnout, jakou prioritu toto přerušení má a co se musí provést. Pokud má přerušení vysokou prioritu, musí se běžící procesy pozastavit a systém se musí zabývat přerušením a jeho požadavkem – v tomto případě třeba napsáním příslušného znaku. Jakmile se znak napíše, systém se vrátí k předchozímu procesu.

Linux tedy stejně jako Windows používá tzv. monolitickou architekturu stavby jádra (viz rámeček napravo). Nicméně i tak může jádro připojovat dynamické moduly. Ty mohou rozšiřovat stávající moduly, nebo je zcela nahrazovat.

Linuxové jádro má jednotné rozhraní jak pro systémové knihovny, tak pro vstupy ze strany uživatele. Rozhraní pro systémová volání přitom hrají důležitou roli při práci s procesy. Pomocí speciálních systémových volání může totiž uživatel přímo ovládat procesy, které běží v jádře.

OS X: Využívá sílu dvou jader

Název operačního systému Mac OS X vznikl zkrácením akronymu XNU – X is Not Unix. I když to není tak úplně pravda. Jádro Applu totiž kombinuje části Unixu a další část pochází z tzv. Mach projektu, což je klasická ukázka mikrojádra (viz text vpravo). Apple nepoužívá Mach jako takový, ale využívá jej jen pro zasílání signálů, tedy pro efektivní komunikaci jednotlivých částí v rámci jádra. K tomu si ještě přidejte XNU Code, který pochází z unixového projektu založeného na FreeBSD. Tato část se stará o správu uživatelů, zpracování signálů a kompatibilitu POSIX. Díky tomu může většina programů ze světa Unixu fungovat také pod Mac OS X.

Velmi důležitou komponentou v Machu je také I/O Kit, tedy vstupně-výstupní systém. Zde je velký rozdíl OS X oproti Linuxu nebo Windows. I/O Kit totiž funguje jako další vrstva mezi hardwarem a zbytkem systému. Najdete zde standardizované modely ovladačů, z nichž jsou odvozeny speciální ovladače jednotlivých komponent. Ty zajišťují vysokou stabilitu a velmi dobrý výkon.

Kromě základních služeb jádra nabízí OS X i rozšíření jádra. Díky tomu je možné systém dynamicky rozšiřovat. Často je proto označováno toto jádro jako hybridní, odborníci jej však označují jako monolitické.

Procesy: Chráněny podpisem

Důležitým úkolem jádra je řízení procesů. Přitom nejde jen o správné přidělení priorit, ale také o ochranu procesů. Při běžném řešení ve Windows jsou procesy ovládány přes Win32-API. Pokud se podíváme detailně do jádra, uvidíme, že řízením se zabývá NTOS. O přístup jednotlivých procesů k objektům jádra se stará tzv. Handles. Procesy díky tomu třeba smějí vytvářet nové procesy. Ve Wordu (proces 1) můžete vytvořit nový dokument (proces 2). Standardní model Windows totiž věří, že Word si svůj nový dokument zase uklidí. Rodičovský proces má totiž plnou kontrolu nad potomky.

Standardní procesy ovšem obsahují díry, na základě kterých je možné získat přístup k počítači. Pokud se podaří hackerovi získat přístup k procesu, který má práva administrátora, může neomezeně alokovat prostor, přistupovat k souborům, měnit je a také spouštět své vlastní procesy. Stanou se neomezenými vládci systému.

Vista proto začala práva procesů výrazně omezovat. Žádný proces nesmí být spuštěn s právy administrátora bez potvrzení uživatelem. Již se tak nemůže stát, že by hacker spustil proces, kterým by ovládal počítač. Navíc jádro systému je chráněno ještě zvlášť a ani administrátor nemůže mít k chráněným procesům systému přímý přístup.

Díky této metodě může být třeba kodek použit při přehrávání filmu jako chráněný proces a jeho spustitelná část musí být digitálně podepsaná. Chráněné procesy jsou krásným příkladem toho, jak Microsoft využil starou architekturu pro vyřešení nových problémů.

INFO

Přehled architektur jádra

MONOLITICKÉ

Velké jádro slouží pro všechny úlohy – taková je idea monolitického jádra. Jádro je zodpovědné za řízení procesů i správu paměti. Stará se i o komunikaci mezi procesy a poskytuje funkce pro ovladače. Obsahuje také podporu hardwaru. Windows, Linux a také OS X patří právě do této kategorie.

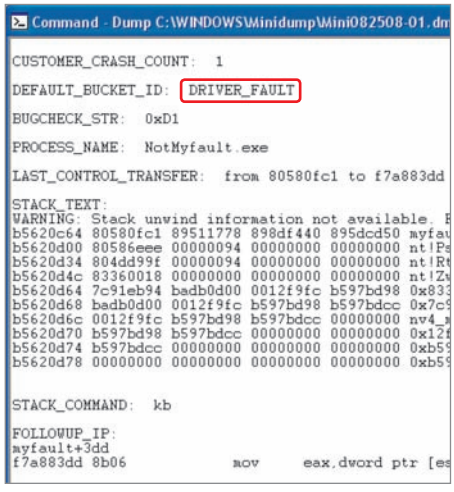
MIKRO

Jedna chyba v jádře dokáže položit celý operační systém. Mikrojádro je proto obzvláště malé, aby v něm nebyl prostor na chyby. Vzhledem k tomu, že jádro musí obsahovat mnoho funkcí, bývá rozděleno na více modulů, z nichž ale jen jeden běží v kernel modu. Typickým příkladem je Mach, což je součást OS X. Jinak se mikroarchitektura v domácích počítačích neuchytila.

HYBRIDNÍ

Kombinace monolitického jádra a mikroarchitektury jádra se nejčastěji označuje jako hybridní jádro. Možnosti samotného jádra jsou rozšiřovány pomocí dynamických modulů. Nahrávání dynamických modulů sice používá OS X i Linux, ale ne do takové míry, aby se jádro mohlo označit jako hybridní.

Řízení procesů v Linuxu a Mac OS X je podobné jako ve Windows. Rodičovské procesy mají plnou kontrolu nad potomky. Zcela chráněné procesy jádra, tak jako to má Vista, ale u těchto systémů chybí. Není divu, neboť Microsoft tuto technologii používá hlavně kvůli Digital Rights Managementu (DRM). Kdo chce tedy na počítači pracovat jako skutečný root, toho Vista



Windows Debugger: Provádí analýzu obrazu paměti a ukazuje, co způsobilo pád systému. V tomto případě to byl ovladač v jádře.

neuspokojí. Bude muset sáhnout po Linuxu nebo OS X, ve kterých může mít plnou kontrolu nad procesy.

ASLR: Maskované adresy

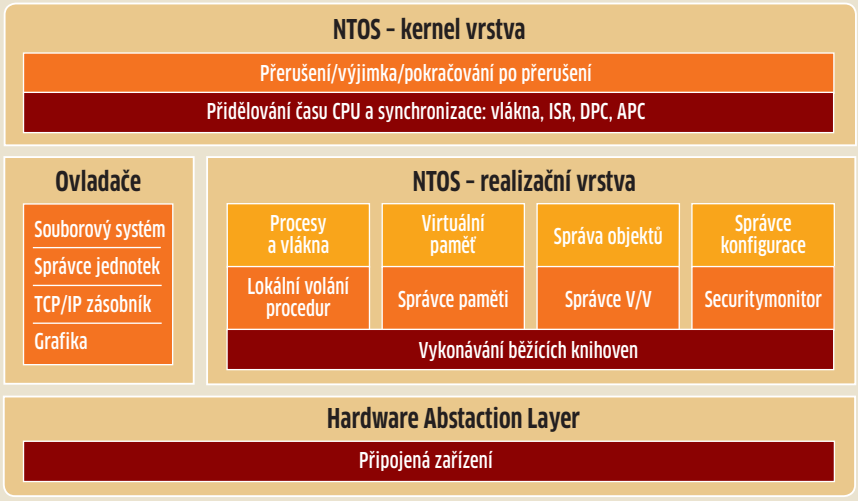
Současné procesy mají 64bitové adresování, některé bity jsou ovšem vyčleněny pro speciální úkoly. Třeba od dob Pentia 4 nebo AMD64 je přítomen NX bit (Non eXecutive bit), což je tzv. bit nespustitelnosti. NX bit je v hlavičce každé stránky operační paměti a operační systém může zvolit, zda je možné tuto stránku vykonat jako spustitelný kód. Pokud se program pokusí spustit stránku, u které je nastavený NX bit, dojde okamžitě k ukončení procesu.

Dokud neexistovala tato ochrana, měli hackeři mnoho cest, jak spustit svůj škodlivý kód v počítači. Stačí vyvolat přetečení paměti v Internet Exploreru, a do paměti, kam už by proces neměl mít přístup, hackeři umístí svůj kód. Pokud se pak systém podívá na tuto stránku v paměti, spustí tím kód a hacker může ovládnout počítač.

Microsoft se snažil zabránit možnému přetečení paměti částečně již v XP SP2, ale mnohem lepší ochranu implementoval až do jádra Visty. Jedná se o funkci Address Space Load Randomaization (ASLR). Dokud Windows tuto funkci neměla, ukládaly se knihovny během startu systému vždy na stejné místo. Tohoto faktu využívali hackeři, protože přesně věděli, co se kde nachází.

Jádro Windows

Soubor NTOS je samotné srdce kernelu. Logicky je rozděleno na dvě části. Jedna zvláštnost: ovladače mohou kvůli vyššímu výkonu přistupovat přímo na hardware.



Funkce ASLR tento řád narušila. Při každém startu systému se mění místa v paměti a malware nemá možnost vystopovat, jaký kód a na jakém místě se v paměti nachází. Pro zamaskování používá správce paměti 256 odlišných adres. DLL knihovna se tak nachází na zcela náhodném místě v paměti. Tento sofistikovaný a vyzdvihovaný systém byl však bohužel prolomen.

Celá řada linuxových distribucí, jako třeba ty založené na jádře Gnetoo, má funkci ASLR implementovánu. Jenže ve standardní konfiguraci není funkce aktivována, aktivaci musí provést uživatel ručně. V současném OS X je funkce ASLR přístupná jen pro vybrané knihovny, kompletní integrace do jádra prozatím chybí.

Hororem pro operační systémy jsou rootkity, které se schovávají do kernel modu. Obvyklé bezpečnostní programy nemají možnost se s nimi dostat do křížku, neboť rootkity se zavádí na stejné úrovni jako samotné jádro. Škodlivý rootkit se do systému obvykle dostane jako ovladač v kernel modu, proto může pozměňovat samotný kernel a také jiné ovladače. Díky tomu je skutečně neviditelný.

Kontrola integrity: Zabezpečení kódu

Prostředek pro boj proti rootkitům přináší Vista v podobě Kernel-mode Codesign (KMCS). To znamená, že do jádra se dostanou jen digitálně podepsané ovladače zařízení. Většina ovladačů dostává podpis přes WHQL (Windows Hardware Quality Lab), vývojáři si však svůj kód mohou podepsat i sami. K tomu ale potřebují platný certifikát. Autor ze svého kódu vytvoří hash, který podepíše svým privátním klíčem a opatří certifikátem, a připojí certifikát i podepsaný hash. Když se Windows pokusí nahrát ovladač, ověří si pomocí podepsaného hashe, zda patří ke kódu, který chce zavést do systému.

Podepsané ovladače zabrání vstupu rootkitů

Windows také kontrolují, zda je podpis určen pro správnou verzi Windows a zda je certifikát platný pro danou modifikaci. Ovladač tak může být podepsán pro 32bitovou Vistu, v 64bitové Vistě už však podpis platit nebude.

Moduly do jádra mohou být podepsané i pro Mac OS X a Linux. Stejně tak to funguje i pro ovladače, ale tyto operační systémy nemají žádnou rutinu, která by podpisy zkoumala a ověřovala.

MMCSS: Preferuje video

Windows žonglují s několika desítkami procesů, které běží na počítači. Tyto procesy běží paralelně vedle sebe a dělí se o čas procesoru. Na běžném počítači bývá kolem 60 až 80 procesů, které se postupně střídají o čas na procesoru. Jakmile jim uběhne vymezený čas, musí počkat, než si jej vyplýtvají i všechny další procesy. Díky tomu můžete sledovat film, přičemž na pozadí systému běží antivirový skener a vy si ničeho nevšimnete.

INFO

Ladicí nástroje pro Windows

WNDBG

Objeví-li se modrá obrazovka, Windows uloží obsah paměti. Abyste ji mohli analyzovat, potřebujete nástroj, jakým je třeba WindDbg. Najdete jej přímo na webu Microsoftu.
Info: www.microsoft.com/whdc/DevTools/Debugging

NOTMYFAULT

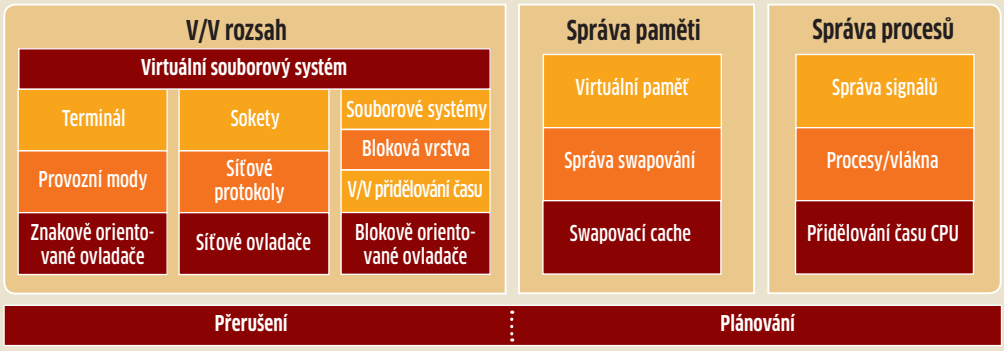
NotMyFault je nástroj na zkoušení stability. Záměrně generuje chyby ve Windows a snaží se systém shodit. Bezvadnému systému by neměl nic udělat. Při experimentování buďte opatrní!
Info: http://technet.microsoft.com/en-us/sysinternals/bb963901.aspx

PROCES EXPLORER

Řízení procesů je jedním z hlavních úkolů operačního systému. Jaké procesy systém spravuje a jakým způsobem, to dokáže zobrazit Process Explorer. Zobrazí veškeré informace, které běžný Správce úloh zobrazit nemůže.
Info: http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx

Linuxové jádro

Linuxové jádro se stará o řízení procesů a také správu signálů. Na nejnižší úrovni jádra jsou funkce pro přerušení a řízení procesů.



Vista jde v tomto ohledu ještě dále se svou službou MMCSS, která upřednostňuje přehrávání multimediálního obsahu. Vše má na starosti knihovna Mmcss.dll, která se nachází v adresáři Windows\System32. Ta obsahuje funkce, které dokáží upřednostňovat vlákna (threats), která mají na starosti přehrávání multimédií. Knihovna podporuje celkem 32 různých priorit. Označení priority nabývá hodnoty 0–31, přičemž pro multimediální vlákna se dosazuje priorita 27. Vlákna s prioritou od hodnoty 16 jsou určena pro běh v reálném čase a jiné procesy je nemohou přerušit. Služba se zároveň stará o to, aby u jednotlivých programů nedošlo k vyhladovění. Z toho důvodu dokáží Windows při aktivované MMCSS vyhradit 1 % času CPU i pro jiné úlohy. Pokud by se aplikace třeba zacyklila, nevedlo by to k pádu systému.

Linux nabízí ještě jemnější plánování priorit. Priority programů mohou nabývat hodnot od 0 do 99. Kdo si chce tedy detailně vyhrát s prioritou každého procesu v systému, ten má širší možnosti než uživatel Windows.

Vrcholem je ovšem Mac OS X. Jeho přidělování priorit procesů má 128 možných hodnot. Navíc má v rukávu dalším trumf – umožňuje totiž aplikaci přidělit přesné procento výpočetního času CPU.

V/V: Vyšší priorita

Co se hodí během přehrávání filmů, to může rozzuřit při práci s více programy. Třeba když se ve Windows XP rozběhne defragmentace na pozadí. Je sice hezké, že po pár hodinách bude harddisk o něco málo rychlejší, ale co z toho, když není možné pracovat v Outlooku?

Pokud jsou priority V/V procesů aktivní, nic takového by se dít nemělo. To funguje třeba ve Vistě. Zde defragmentace probíhá automaticky na pozadí v okamžiku, kdy uživatel nic nedělá. Jakmile uživatel zase začne být aktivní, defragmentace se okamžitě přeruší. Vista umožňuje nastavit pět úrovní priorit. Střední je označena číslem 3 a je to výchozí hodnota po spuštění aplikace. Nižší úrovně mají aplikace, které běží na pozadí systému, a vysokou prioritu mají kritické aplikace. Úroveň ovlivňuje i to, jak aplikace může hospodařit s pamětí. RAM je totiž rychlá, ale vzácná. Pokud jí má počítač jen 1 GB, tak se aplikace s nízkou prioritou odklízí na pomalý disk, zato ty s vysokou zůstávají v paměti.

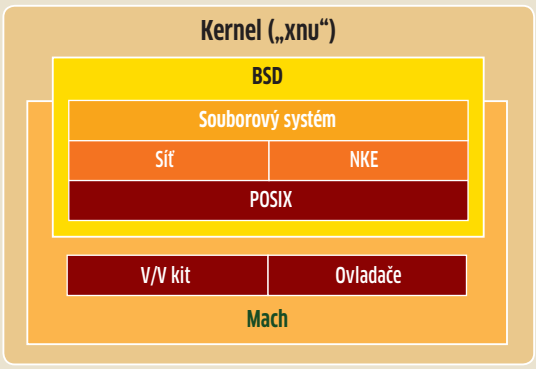
Nejvyšší prioritu musí mít vstupně-výstupní operace. Pokud uživatel hýbe myší, nemůže se pohyb kurzoru po obrazovce provést až v okamžiku, kdy Excel konečně vykreslí graf. Stejně tak není možné pozdržovat pakety na síťové kartě, které přicházejí jeden za druhým ve vysoké rychlosti. Vyrovnávací paměť síťové karty je totiž velmi malá.

Užitečné je, že Vista může rezervovat určité pásmo pro V/V operace. Díky tomu může přehrávač médií přistupovat k DVD mechanice během přehrávání filmu, přesně podle potřeby, a žádná jiná aplikace mu nemůže odebrat přístup k mechanice, aby bylo přehrávání plynulé a bez chyb.

Pro Vistu je upřednostňování vstupně-výstupních operacích velkou novinkou,

Jádro OS X

Jádro OS X vytvořil Apple spojením dvou zdrojů. Používá funkce unixově orientovaného subsystému a mikrojádro Mach.



OS X a Linux mají tuto funkci již velmi dlouho. OS X má tuto funkci zakořeněnou hluboko v architektuře, Mach ji používal při posílání signálů. V linuxovém jádře funkci najdeme od verze 2.6.

Adresace: Dynamicky

32bitové procesy mají ve Windows jedno zásadní omezení při adresaci. Bez speciálních úprav totiž dokáže kernel Windows adresovat jen 2 GB. To vede k problémům, pokud je potřeba použít větší množství paměti. Až do příchodu XP musel správce paměti vědět, kolik místa budou jednotlivé komponenty potřebovat. Problém: Cache narážela velmi brzy na limity, zatímco zásobník měl ještě velké množství prostoru, který však zůstal nevyužitý. Ve Vistě je adresový prostor jádra již zcela dynamický. Správce paměti dynamicky alokuje tolik místa jakékoliv součásti, kolik pro svůj běh potřebuje. Díky tomu se zlepšila i práce s virtuální operační pamětí a nedochází k plýtvání. Každý dostane v paměti jen tolik místa, kolik skutečně potřebuje.

Linux a Mac OS X nemají žádné striktní omezení. Jednotlivé komponenty nemají vlastně žádné hranice. Na rozdíl od Windows nemají pevně danou velikost pro ovladače nebo pro jádro.

Chyby ve Windows jsou proto na denním pořádku. Většinou ale nejde o chyby operačního systému, na vině jsou programy. Tyto potíže pak musí systém vyřešit. Jeden příklad: Aktualizace odstraňuje pouze jednu chybu, ale zaplácává díry na více místech. Co se stane, když je aplikace po polovině aktualizace přerušena? Vše se ještě zhorší. Nejhorší je situace, pokud jde o aktualizaci přímo Windows nebo virového skeneru. Aby se tomu zamezilo, obsahuje systém sledování změn, a v případě potřeby je možné využít funkci Obnovení systému. Ta dokáže vrátit provedené systémové změny zpět. Rovněž si ukládá obraz funkčního obrazu, takže pokud Windows přestanou startovat, stačí před zaváděním jádra stisknout klávesu F8 a zvolit »Poslední známá funkční konfigurace«.

```
# CONFIG_ACPI_DEBUG is not set
CONFIG_ACPI_EC=y
CONFIG_ACPI_POWER=y
CONFIG_ACPI_SYSTEM=y
CONFIG_X86_PM_TIMER=y
CONFIG_ACPI_CONTAINER=m
CONFIG_ACPI_SBS=m

#
# APM (Advanced Power Management) BIOS
#
CONFIG_APM=m
# CONFIG_APM_IGNORE_USER_SUSPEND is not
CONFIG_APM_DO_ENABLE=y
# CONFIG_APM_CPU_IDLE is not set
# CONFIG_APM_DISPLAY_BLANK is not set
# CONFIG_APM_RTC_IS_GMT is not set
CONFIG_APM_ALLOW_INTS=y
# CONFIG_APM_REAL_MODE_POWER_OFF is not
```

Linuxové srdce: Operační systém má jádro zcela přístupné. Právě se díváte na konfiguraci jádra verze 2.6.19.

KTM: Podchycení pádu

Pokud chce aplikace provést celou řadu po sobě následujících operací, které by měly být provedeny najednou, nazývá se tento postup transakce. Systém může k provedení transakce využít handle KTM (Kernel Transaction Manager) a DTC (Distributed Transaction Cordinator), anebo jen KTM použít přímo a změny v souborech nebo klíčích registrů asociovat s transakcí. Pokud vše dopadne úspěšně, transakce se provede a přijmou se změny, které aplikace požadovala. Pokud ne, může se aplikace bez jakékoliv chyby vrátit do výchozího stavu. Až do okamžiku, než proběhnou všechny operace transakce, se žádné změny neprojeví. Vzniknou až naráz po dokončení. Díky tomu ani ostatní aplikace nevidí provedené změny, dokud není transakce dokončena. Transakce je totiž atomická činnost, to znamená, že musí být provedena celá najednou. Typickou transakcí je třeba aktualizace aplikace. Musí se změnit klíče v registrech a nahrát nové soubory. Po změně klíčů ale nesmí dojít k přerušení. Buď se tedy provede vše, nebo nic.

Také Mac OS X a Linux pracují v jádře s transakcemi. Uživatel při vykonávání transakce vůbec nic nepostřehne, až do té doby, dokud něco neselže. Ani v jednom systému ovšem nemá neprovedení transakce vliv na stabilitu operačního systému, protože v případě chyby se jen oznámí, že transakce se nezdařila.

SHRNUTÍ: Windows rozhodně neztrácí krok za Linuxem ani Mac OS X. Microsoftské operační systémy jsou sice velmi mladé, ale jejich architektura pochází ze silného a staréhо světa Unixů. Obzvláště Vista je ukázkou toho, jak velmi staré bezpečnostní řešení a postupy se hodí do moderního operačního systému. Bohužel řada těchto funkcí funguje jen v 64bitovém světě, a ne pod XP. Linux a OS X navíc ztrácí třeba v absenci ASLR, i když už byl tento sofistikovaný systém prolomen. Windows jsou stále oblíbeným operačním systémem hackerů i malwaru, protože se do tohoto systému mohou dostat snadněji než do Linuxu a Mac OS.

Microsoftské operační systémy také mají tu nevýhodu, že s sebou táhnou těžkou kouli na noze. Vychází totiž ze starých systémů a neustále se musí zachovávat značná část zpětné kompatibility. O něco modernější je OS X, především díky komunikaci v jádře, Mach a I/O Kit jsou zárukou vysokého výkonu; zde má Vista co dohánět. Pro Linux hovoří jeho otevřenost: každý si může systém, a to včetně jádra, přizpůsobit – výhoda pro toho, kdo o takovou možnost stojí.

AUTOR@CHIP.CZ

INFO

Windows 7, 8, 9...

Nástupce Visty už je známý, a Microsoft dokonce uvolnil první beta verze. Windows 7 budou v prodeji v roce 2010. Microsoft měl tedy příliš málo času na to, aby v systému provedl zásadní změny. Ani se netají tím, že pracuje na úpravě architektury Windows. Dva projekty ukazují vizi budoucnosti. **Singularity** by měla být Windows bez modrých obrazovek. Jedná se o systém složený ze tří klíčových funkcí: Software Isolated Processes (SIP), mikrojádro a kanály. Mikrojádro se bude starat jen o funkce jádra, jako je správa paměti, správa procesů, řízení kanálů, přidělování času a V/V operace. Všechny ostatní funkce budou obstarávat externí moduly připojené k SIP. Komunikaci mezi jádrem a SIP stejně jako mezi jednotlivými moduly navzájem budou řešit kanály. **Midori** se dívá do daleké budoucnosti modulárních jader. Jen nejhlubší části jádra budou naprogramované v jazyce C nebo v asembleru. Ostatní komponenty budou realizovány pomocí microsoftského .NET. Výhoda: Windows lépe poběží na různých platformách, jako jsou netbooky, PDA nebo mobilní telefony.