11/99

Linuxové noviny

Úvodní slovo

Pavel Janík ml., 31. října 1999

Minulý rok jsem ihned po skončení veletrhu INVEX dostal angínu, a tak jsem hned měl o čem psát. Tento rok byl pro mne sice INVEX ještě náročnější než kdy jindy, ale vše dobře dopadlo. Dokonce i angína se mi vyhnula velkým obloukem. A tak nevím, o čem psát.

Ale abych se ještě vrátil k letošnímu INVEXu – jeho součástí byla poprvé akce nazvaná LinuxHall. O tom, zda byla úspěšná nebo nikoli si musíte udělat názor sami, ale podle mého názoru úspěšná byla. A nejen podle mého názoru. Každý, s kým jsem měl tu čest hovořit, byl z konání takové akce nadšený. Jediným problémem byl nedostatek místa, ale to se u takové akce a navíc u jejího prvního ročníku (a nikdo o dalších ročnících ani nepochybuje) dá čekat. Z úst některých kolegů dokonce zněly názory jako: „Ty chudáky systémové integrátory by měli dát někam jinam, vždyť jim tam nikdo nechodí“.

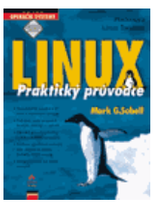
Na LinuxHall jsem se také seznámil s několika zajímavými lidmi, které jsem měl možnost navštívit i po skončení INVEXu, a tak se můžeme snad již v brzké době dočkat podstatného upgradu našeho serveru odysseus, který nese veškerou zátěž serverů ftp.fi.muni.cz a kompletní linuxové konference i konferencí dalších.

A nyní již k aktuálnímu číslu Linuxových novin. Jistě si všimnete, že toto číslo je naplněno články o SQL a vůbec obecně o databázích. To díky Karlu Žákovi, který na svá bedra vzal povinnost navrhnout, realizovat a posbírat veškeré články na toto téma. A že umí pobízet autory k činnosti jsem poznal i na své kůži :-) Ještě jednou díky. Kéž by takových lidí ochotných věnovat kousek svého času i pro ostatní bylo více.

Přeji příjemné čtení a napište nám, jak se vám nové číslo líbí. ■

Recenze: Linux — Praktický průvodce

Oto Buchta



Dostala se mi do rukou již druhá „těžkotonážní“ knížka o Linuxu z dílny vydavatelství ComputerPress z edice Operační Systémy. Tentokrát nemá nálepku PROFI jako LDP, ale je určena širokému okruhu čtenářů. A nenapsal ji nikdo menší než Mark G. Sobell, autor dnes již proslulého „A Practical Guide to UNIX“. Velmi jsem se těšil a Mark mne nezklamal.

Byl jsem zvědavý, zda budu moci tuto knihu doporučit namísto klasické UNIXové „bible“ pánů Brodského a Skočovského, ale není tomu tak.

Nemohu přesně říci, na koho je tato kniha orientovaná. Vedle podrobného popisu funkce Netscape Navigatora Gold (včetně obrázku) je zde návod jak odlaďovat vlastní programy v C. Myslím, že Mark chtěl ukázat Linux (nebo UNIX?) v celé své šíři a možná i někoho kapánek postrašit, neboť bych rád viděl, co asi řekne sekretářka na kapitolu Programovací jazyk příkazového procesoru. Každopádně ale lze říci, že kniha je opravdu praktickým průvodcem obyčejného uživatele po světě Linuxu.

Celá kniha je rozdělena na dva hlavní oddíly. První pod názvem Operační systém Linux — 600 stran textu a obrázků a výpisů, ve kterých autor ukazuje Linux ze všech hlavních pohledů: z pohledů správce, programátora, sečtělejšího uživatele i obyčejného BFU.

Jako výborný nápad se mi jeví zakončení každé z 15 kapitol souhrnem a několika opakovacími cvičeními. Tímto si čtenář, který si vezme knihu jako učebnici, může zrovna vyzkoušet, zda vše správně pochopil. Chybí sice výsledky, ale není to na škodu.

5 kapitol, plných 250 stran je věnováno příkazovým procesorům bash, tcsh a zsh a 100 stran editorům vi a emacs. Z toho IMHO plyne, že hlavním autorovým záměrem bylo naučit lid obecný využívati hlavních výhod operačních systémů UNIX — efektivní a automatizovatelná práce v terminálu v příkazovém procesoru.

Tuto doměnku potvrzují dva fakty. Prvním je na váhu knihy poměrně strohý popis systému X Windows směřující k tomu naučit uživatele spustit si xterm. Druhým a mnohem pádnějším je celý druhý oddíl této knihy, Obslužné programy. Na 250 stranách zde naleznete referenční příručku k 87 programům včetně názorných příkladů. Od cd a cp až po pine či gcc.

Samozřejmě ale žádné dílo není dokonalé. Co mi tedy v knize chybí a co mi na ní vadí? Bez připojení pevnou linkou na Internet je tato kniha jenom poloviční. Na svém webu má Mark mnoho dalších rozšiřujících a upřesňujících informací včetně odkazů, které v knize vysloveně chybí. Myslím, že by si kniha určitě zasloužila přiložené CD-ROM právě s Markovými stránkami.

Kapitola o programovacích nástrojích je podle mne velmi úzce zaměřena na jazyk C. Myslím, že mohl autor dodat, že vedle překladačů existují i velice povedené interprety, které jsou v poslední době čím dál populárnější. Samozřejmě mám na mysli Perl a Python. Kniha je také psána jako oslava Linuxu. V celé knize jsem nenašel ani malou zmínečku o tom, že by bylo něco uděláno nedobře nebo ne úplně kvalitně. Všechno je optimální a geniální a tak. Člověku, který některé chyby vidí, to pak připadá více než podivné. Největší problém ale vidím v zastaralosti. Výsledkem kompilace je soubor a.out, popis Xů končí u fvwm, jádra řady 2.0 jsou oslavována atd. Ale to je, bohužel, pochopitelné vzhledem k době vzniku publikace. V tiráži se můžete dočíst Original copyright: (C) 1997 by Mark G. Sobell. Většina informací je sice pořád aktuální, neboť hlavní myšlenka UNIXu bude moderní i za několik desetiletí a základní programové vybavení se výrazně nevyvíjí, ale to okolí, které je kolem, citelně skřípe. Myslím si, že už konečně dozrál čas, aby buď byly při překladu linuxové literatury používány dodatky s vysvětlením, že teď už to tak není, že nyní je svět Linuxu úplně jinde a nebo aby se dala dohromady parta li-

Linuxové noviny připravuje a vydává České sdružení uživatelů operačního systému Linux

dí v této republice a začala psát knihy o našem oblíbeném operačním systému.

ComputerPress se poučil

Již na první pohled je zde vidět posun oproti LDP. Vazba vypadá bytelněji a lesklý povrch s krásným letícím tučnákem knize svědčí více.

Ale hlavní posun ze strany vydavatelství spatřuji v překladu. Exces s LDP se neopakoval a překlad se jeví býti vysoce zdařilým. Je už méně odporný a je to vidět i na takových maličkostech, jako je například nahrazení slengové „utilitky“ za „obslužný program“. Chyby, které jsem objevil, jsou pouhé překlepy, které i nezkušený laik lehce odhalí. Jedinou výjimku tvoří Grafické uživatelské prostředí Aftostep :-)

Ale mám i námitky a připomínky k sazbě. Na straně 578 je obrázek jaderného konfiguračního prostředí `xconfig`, který se nevztahuje k textu kapitoly, do kterého je vsazen, nýbrž ke kapitole předcházející. Je škoda, že některé výpisy na STDOUT jsou vloženy jako text a jiné jako zarámovaný obrázek. Je to pak trochu matoucí.

Závěrem

Knížka může být užitečnou jak pro úplné laiky, tak pro „profíky“. Je zde sděleno téměř vše, co potřebuje uživatel pracující na Linuxu vědět. Komu tuto knihu doporučit? Dalo by se říct každému, ale pokusím se to přece jenom upřesnit. Publikace nepomůže člověku při instalaci a ani neřekne nic podnětného administrátorovi systému. Programátor aplikací se zde také nedozví nic nového. Kdo tedy zbývá? Uživatel. Ten, který chce či musí pracovat na Linuxu a neví či není si jist, co všechno mu tento systém může nabídnout. Pro něj či pro ni se může stát praktický průvodce opravdu dobrým kamarádem. ■

Počítačová bezpečnost a Linux I – zjišťování informací o serveru

Jarda Cmunt

Úvod

Zabezpečení serveru před průnikem je jednou z prvních věcí, které je třeba udělat, chystáme-li se provozovat nějaké veřejně dostupné služby. To samozřejmě platí i v prostředí interních sítí, neboť podle statistik k 80% neautorizovaných průniků dochází od legitimních uživatelů sítě. V tomto prvním díle se budeme zabývat metodami zjišťování služeb běžících na serveru, neboť sbírání informací je obvykle prvním krokem, který útočník podnikne.

Port scan - detekce služeb běžících na vašem serveru zjištěním otevřených TCP/UDP portů

Scanování portů patří mezi základní metody zjišťování informací o vašem serveru. Útočník ho provádí proto, aby zjistil služby běžící na serveru, což mu umožní udělat si představu o snadnosti nebo nesnadnosti průniku (a ev. také názor na admina). Vzhledem k tomu, že tato znalost může průnik značně usnadnit, je vhodné vědět, jakým způsobem port scan funguje, a jak se proti němu lze bránit. Popišme si základní druhy scanů a možnosti jejich detekce.

connect() scan

Je nejjednodušším a také nejstarším druhem scanu. Scanner se jednoduše opakovaným voláním funkce connect() pokouší vytvořit spojení na všechny (ev. pouze zadané) porty oběti. Pokud se pokus podaří, je port označen za otevřený. Vzhledem k tomu, že z hlediska detekce má mnoho společného s následujícím druhem, popíšeme možnosti detekce později. Tento druh scanu se útočník pokusí použít až jako poslední možnost, neboť programy, na které se spojení podaří, obyčejně tuto událost zaznamenají do logu, a pokud ho admin kontroluje, obyčejně ho (zejména na málo vytížených serverech) může zjistit na první pohled.

SYN stealth scan

Je obdobou předchozího, liší se ale v tom, že nenavazuje plné spojení. Podle příslušného RFC793 probíhá navazování TCP spojení následujícím způsobem: žadatel o spojení zašle na server paket s nastaveným flagem SYN (žádost o synchronizaci sekvenčního čísla) a server této žádosti vyhoví zasláním paketu s nastavenými flagy SYN a ACK, což žadatel zpětně potvrdí ACK paketem a tím je spojení navázáno. SYN scan namísto posledního potvrzujícího paketu zašle RST, čímž žádost okamžitě zruší a ke spojení nedojde. To je výhodné, neboť se tím vyhne záznamu v logu. Pokud se tímto způsobem pokusí komunikovat se zavřeným portem, místo SYN+ACK paketu dostane od serveru RST odpověď. Tímto způsobem dokáže jednoznačně odlišit otevřené porty od zavřených.

Problém s detekcí tohoto typu scanu je, jak odlišit pakety jdoucí od scanneru od paketů, kterými legitimní uživatelé navazují se serverem spojení. Způsobů je několik, můžeme použít některý z následujících algoritmů, případně vymyslet nějaký jiný:

Algoritmus 1: pokud z jedné adresy ve specifikovaném čase přijde pokus o spojení na více než určitý počet portů, jde pravděpodobně o scan.

Algoritmus 2: pokud z jedné adresy přicházejí SYN pakety ve stejných časových intervalech (s rozumnou odchylkou), a navíc jdoucí na různé porty, půjde také pravděpodobně scan.

Algoritmus 3: některé porty (ty, na kterých buď nějaké služby provozujeme, nebo se na nich některé často provozované vyskytují) označíme za hlídané, a pokud na určité množství z nich přijdou SYN pakety ve specifikovaném časovém intervalu, zaznamenáme scan.

Protiakce

Pokud zaznamenáme scan z nějaké adresy, můžeme už v jeho průběhu reagovat tak, že na SYN pakety jdoucí na otevřené porty budeme jako odpověď posílat RST pakety, čímž útočníkovi efektivně znemožníme detekci. Tento způsob chování by měl být zachován ještě po nějakou dobu po skončení scanu. Pokud totiž útočníkův scanner zareaguje inteligentně a sníží frekvenci SYN paketů, tak, že se dostane mimo námi definované meze, stále nebude nic detekovat.

Problémy

Všechny tři zde uvedené algoritmy vyžadují pečlivé nastavení. Na méně zatížených serverech (řekněme do deseti spojení za sekundu) si můžeme dovolit poměrně volné nastavení, ovšem při vysoké zátěži si spolehlivé nastavení může vyžádat větší experimentování (hlavně tam, kde se poskytuje větší množství služeb a přistupuje se tudíž na větší množství portů). Zejména v případě, kdy proti útočníkovi podnikáme výše uvedené protiakce, je třeba se vyhnout falešným detekcím, protože by byly pro legitimní uživatele značně frustrující. I v případě pečlivého vyvážení se nelze vyhnout tomu, že některé pečlivě prováděné scany nebudeme schopni detekovat, např. pokud se útočník zaměří pouze na několik nejpoužívanějších portů a bude posílat jeden paket za hodinu, nebo třeba den. Navíc u všech záznamů v logu je třeba mít na paměti, že zdrojová adresa může být padělaná v zásadě dvěma způsoby: útočník sice scanuje ze své IP adresy, ale zároveň posílá stejné pakety s padělanými adresami z většího množství souboru adres (a my nejsme schopni odlišit jeho od falešných), nebo ke scanu používá jiný počítač, do kterého se již dostal.

FIN, Xmas tree a NULL scan

Tyto tři typy scanů využívají chybné implementace TCP protokolu v Linuxu (a dalších operačních systémech). Nejprve se zmíníme, v čem se liší jednotlivé druhy paketů, které se používají pro průzkum. FIN scan používá pakety s nastaveným FIN flagem (normálně žádost o zrušení spojení), Xmas tree nastavuje kromě toho flagy URG a PUSH, a NULL scan ponechává všechny flagy nenastavené. Podle příslušné specifikace je OS povinen odpovědět paketem RST na všechny pakety, které nejsou součástí některého probíhajícího spojení (s výjimkou RST paketů samozřejmě). Bohužel Linux a některé další systémy tyto pakety jdoucí na otevřené porty pouze zlikviduje bez jakékoliv odpovědi, čímž umožňuje snadno detekovat běžící služby.

Poznámka: pokud se podíváte do kódu, který to má na svědomí (net/ipv4/tcp_ipv4.c u 2.2 a 2.3), zjistíte, že implementátoři strukturou kódu následovali strukturu příslušného RFC, a tento test, který je zmíněn na zcela jiném místě, vynechali. Jiné systémy (Windows, OpenBSD) tuto chybu neobsahují.

Detekce je v tomto případě jednoduchá. Pokud zaznamenáme některý z výše uvedených paketů a nedokážeme ho přiřadit žádnému soketu, jednoduše ho zlikvidujeme a zašleme RST odpověď. Falešné detekce jsou možné pouze v případě zbloudilých paketů, kterých se pravděpodobně nebude vyskytovat nijak významné množství. Patch na jádro, který tento problém řeší systémově, je k dispozici. Dále je třeba poznamenat, že pokud zaznamenáme ve stejný okamžik větší množství scanů z různých adres, měli bychom logovat pouze jeden scan, neboť pravděpodobně všechny pakety budou mít stejného původce a pouze jedna adresa nebude padělaná.

Další typy TCP scanů

Těžko vymyslet něco dalšího. Je zřejmé, že i další možné typy scanů budou založené na chybách nebo nekonzistentnosti implementace TCP protokolu. Uriel Maimon uvádí v časopise Phrack č. 49, soubor 15, zajímavý druh scanu, proti kterému jsou novější jádra imunní. Program zveřejněný v tomto článku posílá na každý port dva ACK pakety s různými hodnotami sekvenčních čísel a window a porovnává došlé RST pakety. Pakety jdoucí z otevřených portů mohou od svých původců převzít hodnotu window, případně mohou mít nižší hodnotu ttl, než pakety jdoucí ze zavřených portů. Program jsem nezkoušel se staršími jádry 2.0, ale při pohledu do zdrojových textů je zřejmé, že právě tato část kódu je zde řešena odlišně než v 2.2 a 2.3, takže na 2.0 a ev. starších by mohl tento druh scanu fungovat. Teoreticky lze další varianty odhalit např. pomocí nějakého generátoru paketů a programu tcpdump. Způsob detekce je shodný jako u FIN & spol. scanů.

UDP scany

Scanováním UDP portů často seriózní hackeři opovrhují, neboť postrádá komplexnost TCP scanů, to ovšem neznamená, že by ho nepoužívali. Mnoho často používaných programů stavěných nad UDP se v nedávných dobách stávalo vděčným terčem útoků, zmiňme např. bind. Vzhledem k tomu, že UDP protokol neudržuje žádné informace o spojení, a podle RFC1122 není ani povinen zasílat chybové zprávy, není UDP scanování obecně nejlehčí. Linux sice dodržuje doporučení a chybové zprávy zasílá, přesto jsou zde další komplikace. Popišme princip funkce.

Pokud zašleme nějaký UDP paket s nesmyslným nebo prázdným obsahem na otevřený port, příjemce mu pravděpodobně neporozumí, zlikviduje ho, a nepošle zpět žádnou odpověď (ev. odpoví UDP paketem). Pokud zašleme paket na zavřený port, systém by měl (ovšem nemusí, ale Linux to dělá) zaslat zpět ICMP zprávu PORT_UNREACHABLE, tj. port je nedostupný. Linux navíc limituje množství těchto odpovědí (jak je doporučeno v RFC1812) na 80 za 4 sekundy, a pokud je tento limit překročen, ještě dále zpomalí.

Je zřejmé, že detekovat běžící UDP služby pod Linuxem tedy lze, a detekce scanu je v principu shodná s detekcí TCP SYN scanu, pouze musíme testovat všechny pakety a vzít v úvahu, že kvůli daným omezením na frekvenci odpovědí bude scanner posílat jednotlivé pakety menší frekvencí. To se odrazí v prahových hodnotách, které nastavujeme pro všechny tři uvedené algoritmy.

Další možné způsoby zjišťování běžících služeb:

- RPC scan – zjistí všechny otevřené porty (TCP i UDP) a na všechny zašle RPC NULL příkaz. Pokud jde o RPC službu, lze tak zjistit číslo a verzi programu.
- HTTP scan – snaží se na www server zasílat požadavky na získání cgi skriptů, které jsou známé svou chybovostí.
- další scany založené na vlastnostech jednotlivých aplikačních protokolů.

Možné problémy spojené s detekcí

Je třeba si uvědomit, že v případě port scanu nejde o aktivní útok na náš server (ev. síť), a tomu přizpůsobit reakci. Předně většina detekovaných scanů nebude zaměřena přímo proti vaší síti, půjde spíše o masově prováděné scany podsítě nebo domény, do které patříte. Hacker si pak z výsledného seznamu bude vybírat ty servery, které budou vypadat zranitelně, takže pokud dokážete scanu zabránit, budete mimo jeho pozornost (je ovšem třeba uvést, že pro ty schopnější můžete v tomto případě znamenat výzvu).

Dále je třeba zvážit případné protiakce. Odepření přístupu z detekované adresy bude v mnoha případech znamenat, že ho odepřete úplně někomu jinému, protože hacker neprovádí scan ze svého počítače. Dále v případě, že je skryt za maškarádou nebo NAT, odepřete přístup i mnoha dalším nevinným uživatelům. Protože nejde vlastně o útok, měli byste se omezit pouze na detekci a zabránění vlastnímu scanu.

Scannery

Pokud chcete vědět, co všechno je možné zjistit o vašem serveru po síti, zde jsou odkazy na nejlepší scannery dostupné pro Linux:

(1) – nmap – nejznámější port scanner od Fyodora. Kromě všech uvedených typů scanů provádí také zjišťování typu OS průzkumem jeho TCP stacku a zjišťuje uživatele, pod kterými běží jednotlivé služby (pokud používáte `identd`). Dále dokáže scan maskovat v souboru zadaných adres a nebo použít jako proxy cizí ftp server (což je celkem dlouho známá vlastnost ftp serverů). Ideální nástroj, pokud chcete zkusit pouze čistý portscan.

(2) – SARA – přímý potomek slavného SATANA od Wietse Venemy a Dana Farmera. V port scanu nijak nevyniká, zato vám dokáže poskytnou o vašem systému spoustu informací, které byste raději neviděli. Má celou databázi možných problémů a všechny zkouší na vašem serveru najít. Výhodou je i snadné ovládání přes browser. Doporučuji instalovat spolu s nmapem a sambou, které dokáže při sbírání informací použít.

(3) – nessus je ze stejného soudku jako SARA, disponuje velkým množstvím pluginů a dokáže poskytnout velké množství informací o zabezpečení vašeho systému.

(4) – ve zmíněném 49 čísle vyšel scanner, který implementuje zmíněnou metodu ACK scanování.

Scan detektory

scan_detect – můj vlastní příspěvek k detekci scanů. Výhodou z hlediska detekčních schopností je, že běží přímo v jádře (jako modul nebo součást jádra), takže má přístup ke všem paketům a navíc je schopen snadno na ně reagovat. Mimo jiné řeší i problém s FIN & spol. scany. Zatím je dostupný pouze pro jádra 2.3, verze pro 2.2 se chystá po odladění. Veřejně bude dostupný na mých připravovaných stránkách o počítačové bezpečnosti pod Linuxem (adresa zatím neznámá). Implementuje všechny zmíněné metody detekce.

(5) – scanlogd – celkem jednoduchý detektor z projektu Openwall. I když se ho rozhodnete nepoužívat, doporučuji navštívit jejich stránky, najdete tam zajímavý bezpečnostní patch na jádra 2.0 a 2.2 (mimo jiné non-executable stack), známý lamač unixových hesel John The Ripper a bezpečný POP3 server.

(6) – portsentry – solidní detektor z projektu Abacus. Při neadekvátní konfiguraci může ovšem způsobit mnoho problémů, od odepření přístupu (viz výše) až po náchylnost k DoS útokům přímo proti detektoru via přeplnění logů. Doporučuji zachovat zdrženlivost při konfigurování těchto schopností.

Samozřejmě existují další, ty jsou ovšem součástí komplexnějších balíků. Budeme se jimi zabývat později.

Sbírání podrobnějších informací

Mezi další informace užitečné pro průnik patří znalost konkrétního typu služeb, které na serveru běží, tj. programů a jejich verzí, dále verze systému a jeho příslušnost k určité distribuci. Užitečnost první informace je zřejmá: pokud útočník zjistí, že na serveru běží např. bind verze 4.9.5-P1, snadno na internetu najde exploit přesně na tuto verzi (zde klasický buffer overflow problém), pokud žádný nenajde, může se pustit do studia zdrojových textů programu, a ev. objevit nějakou novou chybu. Druhá informace může být také užitečná, jednotlivé distribuce obsahují různé defaultní účty, služby a nastavení, které může útočník přímo vyzkoušet. Podívejme se na tyto metody sbírání informací a způsoby, jak jim zabránit.

Sbírání bannerů

Většina interaktivních služeb, jako jsou telnet nebo ftp, se před nebo po přihlášení prezentuje sdělením názvu programu, verze a verze systému, což jsou po útočníka cenné údaje. I další služby, které nekomunikují s uživatelem přímo, mohou ve svých odpovědích některé z těchto informací obsahovat. Navíc mnoho administrátorů trpí přílišnou hrdostí na svůj systém, a i když jejich stránky nemají s Linuxem nic společného, můžete na nich najít loga jako Linux (Redhat, Apache…) inside. Zlomyslný hacker je může snadno přesvědčit, že jejich systém není tak skvělý, jak si mysleli. Je zřejmé, že ve správně zabezpečeném systému je třeba nesdělovat alespoň informace o jednotlivých službách a jejich verzích (navíc je vhodné používat takové programy, které byly programovány s ohledem na bezpečnost, skoro ke všemu existuje vhodná alternativa).

U některých programů lze tyto úvodní informace přímo specifikovat. Měly by obsahovat neutrální informace jako "Vítejte na ftp serveru firmy Security & spol., pokud chcete …". U dalších bude nutné zasáhnout do zdrojových textů programu, pokud však vyžadujete vysoký stupeň bezpečnosti, měli byste takové změny udělat, nebo najít alternativní program, kde lze tyto věci konfigurovat.

Použití informačních služeb

Mnoho služeb běžících na serveru je přímo určeno k získávání těchto informací. Mezi ně patří finger (umožňuje získávat informace o jménech účtů, domácích adresářích, shellech, aktivních uživatelích atd.), portmapper (umožňuje získat porty, na kterých běží RPC služby, a čísla programů, které obsluhují), NIS (je obecně považován za špatně zabezpečenou službu s nízkou bezpečností a mnoha problémy, kterou lze nahradit jinými), některé r-služby, chybně zkonfigurované SNMP, vděčným zdrojem informací může být i DNS.

U jediné služby, kterou poskytnout musíte – DNS – byste měli poskytovat stejně neutrální informace, jako ve výše uvedeném případě. U ostatních služeb je třeba zvážit jejich použití a v případě nutnosti použít upravené verze, které budou poskytovat skutečně pouze ty informace, které poskytnout chcete. Služby jako NIS lze nahradit jinými, např LDAP.

Odposlech síťového provozu

Tato metoda patří mezi nejnebezpečnější, neboť většina

služeb komunikuje nešifrovaně a bezstarostně posílá po síti např. nešifrovaná hesla. Některé služby sice hesla šifrují, ale použitý algoritmus buďto používá slabou šifru, anebo odchycená hesla mohou být podrobena slovníkovému útoku, nebo útoku hrubou silou. Má-li hacker možnost instalovat sniffer do místa, přes které jde velká část vašeho externího síťového provozu, bude mít pravděpodobně jednoduchou práci, a může vynechat ostatní metody. Pokud uživatelé používají na veřejných internetových serverech stejná hesla, jako v interní síti (free i komerční služby), což je běžná situace, může během relativně krátké doby posbírat dostatečné množství párů jméno/heslo, včetně administrátorských.

Obrana není jednoduchá. Tam, kde to jde, byste měli používat šifrování s kvalitními algoritmy, např. ssh místo telnetu, šifrovat poštu, která obsahuje citlivé informace. Měli byste mít kvalitní bezpečnostní politiku, která bude přesně definovat způsob zacházení s hesly s ohledem na jejich možný odposlech, a použít některý ze systémů pro jednorázová hesla atd. Pokud používáte internet k propojení dvou privátních sítí, je jednoznačně nutné šifrovat kompletní provoz. I když je pro Linux dostupná technologie PPTP, měli byste použít IPSec, neboť u PPTP bylo popsáno několik způsobů útokú (zejména ve spolupráci s NT). Proti všem používaným heslům byste měli pouštět program John The Ripper (nebo jiný lamač hesel), hesla, která dokáže rozšifrovat před jejich expirací, jsou špatná hesla.

Sociální inženýrství

Zmíníme jen na okraj, neboť se mu lze těžko vyhnout technickými prostředky. Sociální inženýrství je víceméně umění vydávat se za někoho jiného s cílem získat požadované informace. Zřejmě jedinou účinnou metodou je striktní bezpečnostní politika, která bude definovat, jaké informace je kdo komu oprávněn sdělovat. Hesla patří mezi informace, které by měli znát pouze uživatelé sami, přístup k nim nemá ani administrátor. Ten, kdo je zodpovědný za dodržování této politiky, může ověřovat chování uživatelů právě metodami sociálního inženýrství, je dosti pravděpodobné, že uživatel, který heslo prozradí, to po takovéto lekci podruhé neudělá.

Metody, které můžete použít proti vlastním systémům

Nejjednodušší a z hlediska efektivnosti nejhorší je telnet na každý otevřený port. Takto můžete posbírat bannery od interaktivních služeb stavěných na TCP protokolu a komunikujících textově. Patří sem např. telnet, SMTP, POP3, IMAP a HTTP. Efektivnější metodou může být použití programu netcat (standardně v některých Linuxových distribucích), kde můžete sbírání automatizovat pomocí skriptu. Také některé výše uvedené scannery toto dělají automaticky. Pokud chcete vědět, co se skrývá v netextových a UDP paketech, můžete použít některý ze snifferů, které používají hackeři, nebo standardní tcpdump.

Závěr

Je jednoduchý. Měli byste se starat o to, co vše může o vašem serveru kdokoliv zjistit, a pokud možno tomu zabránit a pokusy detekovat. Velmi užitečné je použít alespoň první dva scannery proti vlastním systémům a udělat si tak obrázek o jejich zabezpečení. Pokud hrdě sdělujete, že na vašem serveru kromě httpd běží také lpd, identd, finger, rpc služby, nebo dokonce linuxconf, pouze proto, že se to tak defaultně nainstalovalo, hacker si snadno udělá obrázek o vašich administrátorských schopnostech a zabezpečení serveru.

Příště

Bezpečnost jednotlivých služeb, alternativy k běžně používaným programům. ■

1 Insecure nmao
http://www.insecure.org/nmap/
2 SARA
http://www.arc.com/se.html
3 Nessus
http://www.nessus.org
4 phrack
http://www.phrack.com
5 scanlogd
http://www.openwall.com
6 Abacus portsentry
http://www.psionic.com

SQL aneb jak se domluvit s databází

Jirka Kosek

Dotazovací jazyk SQL je dnes standardem pro komunikaci s většinou databází. Jeho znalost je proto nezbytná pro všechny programátory, kteří vyvíjejí nějaké databázové aplikace. Vhod přijde i pokročilejším uživatelům, kteří používají nestandardní dotazy, na jejichž vytvoření nestačí vizuální dotazovací nástroje. Nakladatelství Grada zaplnilo citelnou mezeru na českém trhu, když ve své řadě Kompletní kapesní průvodce vydalo příručku o SQL.

Již z prvních kapitol je vidět, že autor Milan Šimůnek má s databázemi bohaté zkušenosti. Stručně a srozumitelně zde vysvětluje všechny databázové pojmy, které jsou používány v dalším výkladu. V druhé kapitole se po krátkém historickém ohlédnutí seznámíme s možnostmi jazyka SQL.

Mně osobně se velice líbila třetí kapitola, která se věnuje návrhu datové základny. Pokud sami vytváříme nějakou databázovou aplikaci, je správně navržená struktura tabulek jednou z nejdůležitějších věcí. A po pravdě řečeno, i mnoho zkušenějších uživatelů dělá v této oblasti chyby. Přečtení této kapitoly prospěje každému, kdo někdy vytvářel vlastní databázi.

V následující kapitole jsou probírány pokročilejší vlastnosti jazyka SQL, které už nevyužije zdaleka každý. Dozvíme se, jak vytvářet transakce a jak omezovat přístup uživatelů k databázi. Tím zvládneme dvě techniky, které je vhodné používat pro zajištění bezpečnosti a konzistence dat. Dále je v kapitole stručně zmíněna možnost vytváření procedur a triggerů. Závěr kapitoly využijí zejména ti, kdo pracují s rozsáhlými objemy dat. Dozvíme se zde, jak optimalizovat datovou základnu z hlediska její velikosti a rychlosti přístupu. Jsou rozebrány i možnosti zrychlení provádění dotazů.

Zejména pro začínající uživatele je vhodná pátá kapitola, kde je na množství konkrétních příkladů ilustrováno použití jazyka SQL. Touto kapitolou také končí výkladová část knihy a začíná podrobná referenční příručka.

O referenční části knihy se nebudu nijak dlouze rozepisovat. Do několika kapitol je rozdělen vyčerpávající popis všech příkazů jazyka SQL. Samostatná kapitola je věnována příkazu SELECT, který je nejpoužívanější a jenž má mnoho volitelných částí. Kromě příkazů nalezneme v referenční části i popis funkcí, konstant, datových typů a operátorů.

Mezi nejrozšířenější databázové systémy v Čechách patří MS Access a Oracle. Access je používán zejména pro menší aplikace, oproti tomu Oracle splní i ty nejnáročnější požadavky. Uvítal jsem proto, že v referenční části jsou popsány i odlišnosti těchto dvou produktů od standardu SQL.

Závěrem mi nezbývá nic jiného, než knihu doporučit všem, kteří potřebují pracovat s jazykem SQL. V knížce naleznete vše velmi rychle a její formát vás přímo vybízí k tomu, abyste ji nosili stále s sebou. Tomu je přizpůsobena i vazba knihy, která je šitá, a nestane se vám tedy, že po pár týdnech práce s knihou se rozlepí hřbet a začnou z něj vypadávat jednotlivé listy.

Milan Šimůnek: SQL – kompletní kapesní průvodce. Grada 1999, 248 stran, ISBN 80-7169-629-7.

Recenze původně vyšla v týdeníku Computerworld 18/99. Speciálně pro čtenáře Linuxových novin ji ještě trošku doplním.

Kniha se zaměřuje na standard jazyka SQL a na odlišnosti Accessu a Oracle od standardu. Mezi uživateli Linuxu jsou velice oblíbené databázové servery MySQL a PostgreSQL. Určitě vás tedy napadne otázka, zda má kniha nějakou hodnotu pro uživatele jiných databází než Access a Oracle. Podle mého názoru má. Standardní příkazy jazyka SQL podporované všemi servery jsou v knize popsány dobře. Nadstandardní rozšíření, které přináší například MySQL a PostgreSQL, v knize nenalezneme. Podle mne je však lepší používat je jen v nezbytně nutných případech. Získáme tím kód, který bude snáze přenositelný mezi různými SQL servery. ■

Začínáme s SQL na Linuxu

Pavel Janík ml.

Úvod

Operační systém Linux se již pomalu stává stálicí i na databázovém trhu, a tak již není žádný problém provozovat téměř libovolný databázový server na Linuxu. Můžete si vyzkoušet např. Oracle, Informix, Sybase a další.

Jenže… Kde je tedy problém? Co když náš počítač nepatří zrovna mezi ty nejžhavější novinky a jeho procesor a paměťové moduly (např. Intel 486/8MB) by mu mohlo závidět i leckteré muzeum počítačové historie? Jaké máme možnosti? Oracle či Informix použít nemůžeme, protože tyto databázové systémy mají někdy problémy i na podstatněji výkonnějších strojích (ostatně tyto databáze jsou vytvořeny pro enterprise prostředí, kde také hrají prim). Zbývají nám systémy jako PostgreSQL, MySQL nebo MiniSQL. Servery PostgreSQL i MySQL jsou velmi výkonné, ale i přesto velmi pomalé na naší hypotetické konfiguraci. Podívejme se tedy na server MiniSQL (aka mSQL nebo Minerva).

MiniSQL je komerčním produktem společnosti Hughes Technologies Pty, Ltd. (1). Licenční podmínky jsou benevolentní pro nekomerční subjekty, které mohou MiniSQL kopírovat, šířit, či dokonce modifikovat bez jakýchkoli omezení. Poslední verze s číslem 2.0.11 byla uvolněna 23. srpna tohoto roku.

Instalace

Instalace serveru MiniSQL je ve světě Linuxu poměrně nestandardní. Je sice využito standardních prostředků poskytovaných většinou linuxových distribucí (`autoconf`), ale velice specifickým způsobem. I přesto je instalace poměrně jednoduchá a pokud zvolíte již připravenou binární distribuci ve formátu RPM, je naprosto bezproblémová. Máme-li k dispozici pouze zdrojové texty serveru MiniSQL, musíme je nejdříve rozbalit:

```
tar xfz msql-2.0.11.tar.gz
```

Rozbalené zdrojové texty jsou nyní v adresáři msql-2.0.11 a mají níže popsanou strukturu. V hlavním adresáři jsou soubory BUGS (informace o tom, jak ohlašovat chyby v serveru msql), INSTALL (popis instalace serveru), MSQL_BOOK (o MiniSQL vyjde i kniha :-), Makefile (standardní soubor pro make), README, README.sco, RELEASE_NOTES (informace o aktuální verzi serveru MiniSQL). Vedle těchto souborů jsou v distribuci MiniSQL i adresáře: demos (obsahuje demo-aplikaci – záložky), doc (obsahuje kompletní dokumentaci ve formátu HTML a PostScript), misc, scripts a src. Poslední jmenovaný adresář obsahuje kompletní zdrojové texty serveru a ještě se o něm zmíníme.

A nyní k vlastní instalaci. MiniSQL je možno přeložit na velkém počtu platforem a operačních systémů, a tak obsahuje podporu pro více platforem, která je implementována na adresářovém principu. Každá architektura bude mít svůj adresář, který vytvoříme pomocí příkazu

```
make target
```

Výstup, který bychom mohli obdržet, může vypadat např. takto:

```
Making target directory for Linux-2.3.20-i686

Building directory tree.
        Adding common
        Adding conf
        Adding lang-common
        Adding lite
        Adding makedepend
        Adding makegen
        Adding msql
        Adding regexp
        Adding tests
        Adding tests/rtest.src
        Adding w3-msql
        Adding w3-msql/tests

Adding sym-links
Build of target directory for Linux-2.3.20-i686
complete
```

Nyní bude vytvořen adresář targets a v něm další adresář pro naši architekturu. V našem případě bude mít jméno Linux-2.3.20-i686. V tomto adresáři již budeme mít připraveny zdrojové texty k překladu, a proto se do něj přepneme:

```
cd targets/Linux-2.3.20-i686
```

Dále spustíme konfigurační utilitku, která využívá služeb autoconf k rozpoznání vlastností cílového prostředí:

```
./setup
```

Nyní již jenom zkontrolujeme obsah souboru site.mm, případně upravíme instalační cesty (proměnná INST_DIR), a můžeme spustit vlastní překlad a instalaci:

```
make
make install
```

Mezi tím, než nám doběhne překlad, si musíme rozmyslet, pod jakým uživatelem náš SQL server poběží – pro testování může běžet SQL server pod právy uživatele, který server instaloval, ale pro rutinní provoz bude lépe vytvořit zvláštního uživatele (např. msql), který bude sloužit pouze pro běh SQL serveru.

Konfigurace

Server MiniSQL je možno jednoduše konfigurovat. Veškeré konfigurační informace jsou uloženy v souboru ...instalační adresář.../msql.conf. Konfigurační soubor je rozdělen do sekcí general, system a w3-msql.

Sekce general může vypadat např. takto:

```
[general]
Inst_Dir = /usr/local/
DB_Dir = %I/msqldb
mSQL_User = msql
Admin_User = pavel
Pid_File = %I/msql2d.pid
TCP_Port = 1114
UNIX_Port = %I/msql2.sock
```

Hodnota Inst_Dir ukazuje na adresář, který jsme zvolili při překladu. Pod tímto adresářem jsou schovány ostatní zajímavé soubory a adresáře. DB_Dir ukazuje na adresář, kde jsou uloženy datové soubory serveru. Pokud je ve jméně tohoto nebo i ostatních adresářů uvedeno %I, je tato dvojice znaků nahrazena obsahem proměnné Inst_Dir.

Proměnná mSQL_User obsahuje jméno uživatele, pod kterým poběží databázový server jako takový (viz výše). Admin_User je uživatel, který bude mít povoleno provádět privilegované příkazy (vytváření resp. rušení databází, případně zastavení či reload databázového serveru).

Proměnná Pid_File obsahuje jméno souboru, ve kterém bude v každý okamžik běhu databázového serveru MiniSQL uloženo identifikační číslo procesu hlavního démona msql2d.

Proměnná TCP_Port obsahuje číslo TCP portu, na kterém bude server MiniSQL čekat na příchozí spojení (pokud bude spojení ze vzdáleného serveru povoleno).

Poslední proměnnou je UNIX_Port. Obsahem této proměnné je jméno socketu, který bude použít pro komunikaci s databázovým serverem na lokálním počítači.

Sekce system má stejnou strukturu, ale odlišné proměnné:

```
[system]
Msync_Timer = 30
Host_Lookup = True
Read_Only = False
Remote_Access = True
Local_Access = True
Query_Log = True
Query_Log_File = %I/query.log
```

Proměnná Msync_Timer obsahuje počet sekund, po kterých bude synchronizován obsah databáze na disku s obsahem databáze v paměti databázového serveru.

Proměnná Host_Lookup má podobnou funkci jako hodnota KNOWN v tcp_wrapperu. Pokud je tato proměnná nastavena na hodnotu True a klient, který se připojuje přes TCP port k lokálnímu databázovému serveru nemá v pořádku záznam v reverzní doméně, má bohužel smůlu a nepodaří se mu k databázovému serveru přípojit. Pokud je tato hodnota nastavena na False, připojit se může i ten, kdo tento záznam v pořádku nemá.

Význam proměnné Read_Only je zbytečné vysvětlovat. Pokud je tato proměnná nastavena na True, je server spuštěn v režimu pouze pro čtení, a tedy žádný SQL příkaz nemůže změnit obsah databáze.

Další dvě proměnné umožňují specifikovat režim serveru – server může poskytovat služby pouze lokálním klientům (Remote_Access = False, Local_Access = True), pouze vzdáleným klientům komunikujícím po TCP (Remote_Access = True, Local_Access = False) nebo všem klientům (Remote_Access = True, Local_Access = True). Pro nasazení na webový server je vhodné použít první popsanou metodu.

Poslední dvě volby slouží především pro odlaďování aplikací, kdy je možno zaznamenávat veškeré přijaté SQL dotazy do souboru v následujícím formátu:

```
11-Oct-1999 22:09:45 pavel UNIX_SOCK pokus 95
INSERT INTO tabulka  VALUES (1, 'sloupek1')
```

Formát je zřejmý – v prvním řádku je uvedeno datum a čas položení SQL dotazu z druhého řádku. Dále je zde uveden uživatel (pavel), databáze (pokus) a typ spojení (UNIX_SOCK). Pro reálné nasazení doporučuji tyto volby ponechat vypnuté, neboť poněkud snižují odezvu serveru.

Startujeme

Nyní již tedy známe vše potřebné pro provozování serveru, ale stále ještě nevíme, jak vlastní server spustit. Je to jednoduché – při instalaci databázového serveru byl vytvořen i adresář bin/, který obsahuje několik důležitých programů.

Nejdůležitějším z nich je program msql2d, což je vlastní server. Po jeho spuštění si můžeme ukázat funkci dalších programů z tohoto adresáře:

```
./msql2d
```

Po spuštění server vypíše základní informace:

```
Mini SQL Version 2.0.11
Copyright (c) 1993-94 David J. Hughes
Copyright (c) 1995-99 Hughes Technologies Pty Ltd.
All rights reserved.

 Loading configuration from\
     '/usr/local/msql.conf'.
 Server process reconfigured to accept\
     200 connections.
 Server running as user 'msql'.
 Server mode is Read/Write.

Warning : No ACL file.  Using global
read/write access.
```

Poté je již vše připraveno pro naši práci – server běží a je připraven naslouchat příkazům svého pána – databázové-

```
Server Statistics
-----------------

Mini SQL Version 2.0.11
Copyright (c) 1993-94 David J. Hughes
Copyright (c) 1995-99 Hughes Technologies Pty Ltd.
All rights reserved.

Config file      : /usr/local/msql.conf
Max connections  : 200
Cur connections  : 1
Running as user  : msql
Server uptime    : 0 days, 0 hours, 8 mins, 36 secs
Connection count : 7
Query count      : 0

Connection table :
  Sock Username  Hostname    Database Connect   Idle   Queries
 +----+--------+------------+--------+---------+-----+--------+
 |  5 | pavel  | UNIX Sock  | pokus  |  0H  0M |   0 |      1 |
 +----+--------+------------+--------+---------+-----+--------+
```

Výpis č. 1: Stav serveru

ho administrátora, který má k dispozici silný nástroj. Je jím program msqladmin, který umožňuje komfortní práci s databázovým serverem. Má několik možných argumentů, pomocí kterých je možno např. vytvořit databázi (msqladmin create), zrušit databázi (msqladmin drop), kopírovat či přesunout databázi (msqladmin copy/move). Samozřejmě je možné databázový server restartovat (msqladmin reload) či ukončit (msqladmin shutdown). Zajímavou vlastností je také možnost sledovat aktuální Stav serveru samotného pomocí příkazu msqladmin stats.

Dalšími zajímavými příkazy pro administrátora jsou msqlimport a msqlexport, které umožňují jednoduchý přenos dat mezi textovou a databázovou podobou. A ještě zajímavějším příkazem je msqldump, který ukládá kompletní databázi do textového formátu, pomocí kterého je možné okamžitě databázi obnovit. To je vhodné zvláště pro zálohování kompletního databázového serveru.

Zajímavým nástrojem je také program relshow, který nám pomůže při zjišťování obsahu databázového serveru – může nám prozradit seznam databází na serveru (`relshow`) nebo obsah databáze, tj. seznam tabulek (`relshow database`) či sloupců v tabulce (`relshow database table`).

Přístupová práva

Téměř samozřejmou vlastností databázových serverů moderní doby je možnost správy přístupových práv až na úroveň jednotlivých sloupců v tabulkách. Databázový server MiniSQL je bohužel v této oblasti poměrně pozadu a umožňuje pouze kontrolovat přístupová práva na úrovni jednotlivých databází. Vše se opět děje na úrovni editace souboru (oproti standardním SQL příkazům GRANT a REVOKE) msql.acl:

```
database=pokus
read=nobody
write=pavel
host=*
access=local,remote
```

Pokud soubor `msql.acl` vypadá jako výše uvedený příklad, může uživatel pavel databázi pokus měnit a ostatní uživatelé z ní může pouze číst. Přístup je povolen jak z lokálního, tak ze vzdáleného počítače. Podrobnější dokumentaci k přístupovým právům naleznete v manuálu k serveru.

SQL příkazy

Server MiniSQL není plnohodnotným SQL serverem, a to je nutno při jeho nasazení brát v úvahu. Samozřejmě jsou podporovány jednoduché příkazy CREATE (TABLE, [UNIQUE] INDEX, SEQUENCE), DROP, INSERT, DELETE, UPDATE, SELECT (WHERE, ORDER BY, AND, OR). Nejsou podporovány implicitní funkce jako např.:

```
SELECT MAX(Plat) FROM Zamestnanci;
```

a vnořené příkazy SELECT. Zajímavou se může zdát podpora standardních operátorů LIKE a také vlastní implementace operátorů RLIKE (kompletní regulární výraz) či SLIKE (fonetické porovnávání).

Příkazy samotné je možno zadávat po spojení s databázovým serverem. Spojení je možno navázat pomocí programu msql, která je klientskou částí MiniSQL.

Systémové proměnné

Zajímavou vlastností serveru MiniSQL jsou také systémové proměnné – tedy jakési virtuální sloupce v tabulkách. Systémové proměnné poskytují informace nezávislé na vlastní struktuře a významu dat. Mezi implementované systémové proměnné patří _rowid (jednoznačný identifikátor řádku v tabulce), _timestamp (časová známka poslední modifikace řádku tabulky), _sysdate a _systime (aktuální datum a čas) a další.

Celkové hodnocení

MiniSQL je velice rychlým a jednoduchým databázovým serverem. Jeho rychlost je ovšem vyvážena nekompletní podporou jazyka SQL, a tím i složitější prací. Jednoduchá

administrace je zase protipólem příliš hrubému rozvrstvení přístupových práv (pouze pro databáze, přičemž lepší servery mohou přístupová práva specifikovat až na úrovni sloupců tabulek).

Před jeho nasazením je třeba pečlivě zvážit všechna pro a proti. Rychlost versus nízký komfort při administraci, jednoduchost versus kompatibilita. Ale i o tomto je databázový svět – databázových serverů je bezpočet, jen si vybrat ten nejvhodnější. ■

1 Hughes Technologies Pty
http://www.hughes.com.au/

MySQL databázový server

Jan Pazdziora, adelton@fi.muni.cz

MySQL patří mezi jednodušší databázové servery, jednodušší ve smyslu, že implementuje jistou podmnožinu funkcí „velkých“ databázových serverů, implementuje je dobře a rychle. Mezi jeho přednosti patří rychlost, přenositelnost, dobrá podpora propojení s okolním světem, mnoho vestavěných SQL funkcí. Co chybí jsou transakce, triggery či referenční integrita. Mezi jeho přednosti patří také jeho mostly-free licence.

Domovskou stránkou MySQL je (1). Server i klienti jsou k dispozici jak ve zdrojové podobě, tak pro mnoho platforem i v binární, včetně několika populárních distribučních formátů (například rpm). Překlad ze zdrojových textů je standardní

```
./configure
make
make install
```

a samozřejmě je dobré číst dokumentaci. Překompilovat je nutno, pokud chceme, aby server používal jinou než standardní (ISO-Latin-1) znakovou sadu. K dispozici je jednak standardní ISO-Latin-2, jednak podpora pro české třídění. Tu zkompilujeme pomocí parametru

```
./configure --with-charset-czech
```

Ve dřívějších verzích způsobovalo použití českého třídění chyby v indexech, poslední 3.22 a nejnovější 3.23 už tuto chybu nemají (alespoň jsem na ni nenarazil). České třídění pracuje interně se znakovou sadou ISO-8859-2, pokud chceme použít na klientské straně jinou znakovou sadou, povolíme v souboru sql/convert.cc přístup k překódovacím rutinám a nastavíme překódovávání přenášených textových dat SQL příkazem SET OPTION CHARACTER SET.

MySQL je primárně nástroj pro uložení dat v tabulkách, a pro přístup k těmto datům (změny, výběry) prostřednictvím SQL — Structured Query Language (některé kapacity tvrdí, že to S je Simple). Po instalaci, vytvoření základní databáze a tabulek skriptem scripts/mysql_install_db.sh a spuštění serveru můžeme k připojení použít řádkový klient mysql. Zde pak již zadáváme SQL příkazy a dostáváme odpovědi, buď ve formě tabulky s vybranými řádky, či informací o provedené akci. Server i klientská část jsou vysoce konfigurovatelné, ať již parametry příkazové řádky či konfiguračními soubory. Je tedy možno zadat například port, na kterém server očekává příchozí spojení, či nejrůznější výkonové parametry typu velikosti bufferů. Velmi rozsáhlá a podrobná dokumentace je naším přítelem, pokud chceme server využít opravdu efektivně.

MySQL používá pro uložení dat jednoduchý postup databáze == adresář, tabulka == soubor (resp. více souborů, jeden na data, další na indexy). Tedy

```
$ mysql test
mysql> create table names\
   (id integer, name varchar(20)) ;
```

vytvoří v podadresáři test soubory names.* a s tímto souborem nadále pracuje. To dává například možnost zálohovat pouze ty tabulky, které považujeme za podstatné. Vzhledem k tomu, že v MySQL není k dispozici podpora pro foreign klíče, server se snadno vyrovná i se situací, kdy ho spustíme nad databází (adresářem), kde je polovina tabulek novějších a polovina starších, obnovených ze zálohy (vyrovná ve smyslu, že při spuštění nebude hlásit chybu a tabulky akceptuje v takové podobě, v jaké mu je dáme).

Na MySQL je pěkné, že dává uživateli co největší kontrolu nad daty. Například množství datových typů, pro celá čísla odstupňovaných po jednom bytu, či fakt, že s „dlouhými“ daty (BLOB, TEXT) je ve většině případů pracuje zcela stejně jako s daty typu VARCHAR. Ohromující je také množství vestavěných SQL funkcí na manipulace téměř všeho se vším, či SQL příkaz REPLACE.

Z hlediska standardů MySQL implementuje jistou podmnožinu ANSI SQL92 s mnoha rozšířeními. Odchylky v syntaxi nejsou velké a jsou často vedeny potřebou zapodporovat i funkčnost navíc.

MySQL dovoluje definovat přístupová práva na základně více kritérií – na tabulky, sloupce tabulek, databáze, podle uživatelů či podle strojů, ze kterých se hlásí. Systém práv je relativně složitý, velkým pomocníkem při testování správného nastavení může být utilita mysqlaccess. Dokumentace ale velmi podrobně popisuje algoritmus, který MySQL při přístupu k datům používá, počítačově nadaný člověk by tedy neměl mít s vyladěním pro konkrétní situaci problémy (a nenadanému pravděpodobně postačí, že do databáze test je možno se v defaultní konfiguraci dostat bez hesla). Novější verze MySQL podporují již i příkaz GRANT a REVOKE, informaci o přístupových právech je ale možno přímo editovat i v tabulkách user, db, host, tables_priv a columns_priv v databázi mysql. Pokud budeme měnit záznamy o přístupových právech přímo v tabulkách, je dobré nezapomenout dát posléze mysqladmin flush-privileges, aby server vzal nové nastavení na vědomí.

Vzhledem k tomu, že MySQL nepodporuje transakce, jsou všechny akce okamžitě provedeny, nejsou zaznamenávány do rollback segmentů. K dispozici je ovšem několik úrovní logování změn — ty sice není možno použít k automatickému rollbacku například po pádu počítače, ale jako prostředek pro dohledání, co se s daty stalo, ve většině případů stačí.

Neexistence transakčního zpracování diskvalifikuje MySQL z „velkých“ aplikací, ovšem je zcela vyhovující (a často i výhodou) v hlavní doméně nasazení serveru — databázový server na Web serveru. A zde databáze může být měřeno počtem řádků dosti rozsáhlá. Režie při navázání spojení s databází je velmi malá a připojení tudíž velmi rychlé. Pro typické aplikace, které ve většině případů potřebují udělat pouze connect a 5 selectů je tato vlastnost ideální. Existují rozhraní pro snad všechny rozšířené skriptovací i standardní jazyky, od ODBC, JDBC přes PHP a Perl až po klasické API v jazyce C.

MySQL je vyvíjen firmou TcX AB a není obecně šířen pod GPL (byť vybrané starší verze jsou k dispozici pod GPL). Licence je v zásadě taková, že pokud neprodáváte

produkt, jehož by byl MySQL server součástí, můžete ho v Unixové verzi používat zdarma, tedy i ve vlastní firmě ke komerčním účelům. Klientská část je navíc šířena jako public domain. Můžete samozřejmě podpořit vývoj MySQL zakoupením licence či některého z typů podpory.

Vývojáři jsou velmi přístupní podnětům a patchům, domnívám se proto, že má cenu o MySQL i z dlouhodobého hlediska uvažovat, byť není přímo pod GPL. Přímo srovnat by se pravděpodobně dal s MiniSQL serverem, z něhož ideově vyšel, zde MySQL jasně vede jednak svými možnostmi, jednak aktivní odezvou vývojářů na hlášení chyb. Na druhé straně PostgreSQL přidává transakční zpracování a větší snahu implementovat vše, co ANSI standardy říkají. ■

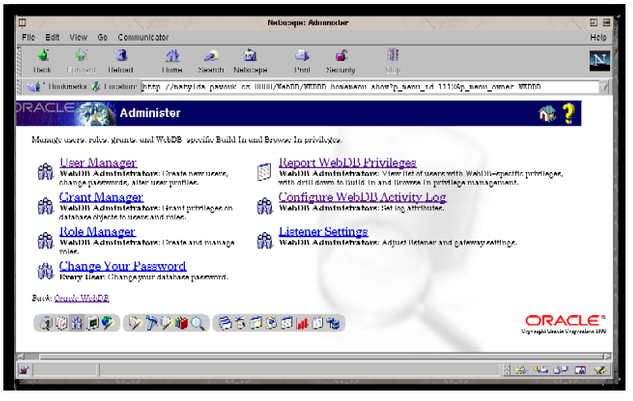
1 MySql
http://www.mysql.com/

Oracle WebDB

Vladimír Náprstek

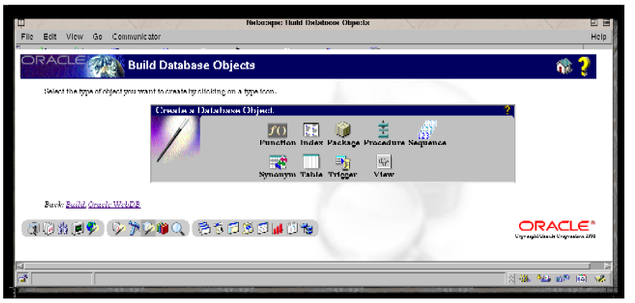
Oracle WEBDB (OW) je nový produkt firmy Oracle. Jedná se o „nadstavbu“ databázového stroje Oracle RDBMS 8.0.x nebo 8i (8.1.5). A k čemu to je? To se dovíte v následujícím textu. Protože již vyšlo několik recenzí (např. v Chipu) a i na www firmy Oracle CZ najdete dost informací, budeme se zabývat hlavně instalací a prvními dojmy.

Takže jenom na úvod, co je cílem WebDB? Pomocí něho můžete vytvářet dynamické www aplikace, kompletně uložené v databázi. Co je velmi, velmi zajímavé, je to, že veškerý kontakt s databázovým strojem probíhá pouze prostřednictvím www prohlížeče (libovolného, umí-li rámy a javaskripty). Tzn. veškerá správa, programování a posléze i využívání hotové aplikace probíhá pomocí Vašeho oblíbeného brousku.



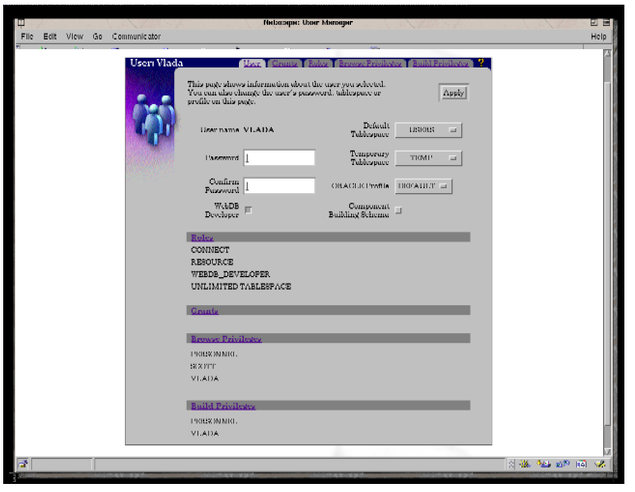
Architektura

Základem je samozřejmě databázový stroj Oracle 8.0.x nebo 8i (8i má být standardní součástí, ale v současné době ještě není verze pro Linux dostupná). WEBDB je package, který se pouze nainstaluje. Pro komunikaci s uživatelem je potřeba ještě nějaký http démon. To může být webdb listener, nějaký WWW server (jako příklad je uváděn Apache nebo IIS) nebo cartridge pro Aplikační server Oracle. Preferovanou možností je využití webdb listeneru, protože ten si při startu vytvoří spojení do databáze a udržuje jej po celou dobu svého života (ve skutečnosti se těch listenerů pustí vícero – jako u Apache). Naproti tomu použití std. WWW serveru znamená jít cestou cgi skriptů. Funkčně je to stejné, jen rychlost by měla být v případě webdb listeneru vyšší, protože odpadá režie s přihlašováním do DB. Prozatím jsme zkoušeli jen webdb listener, takže nemůžeme podat svědectví o rozdílu.



Podporované OS

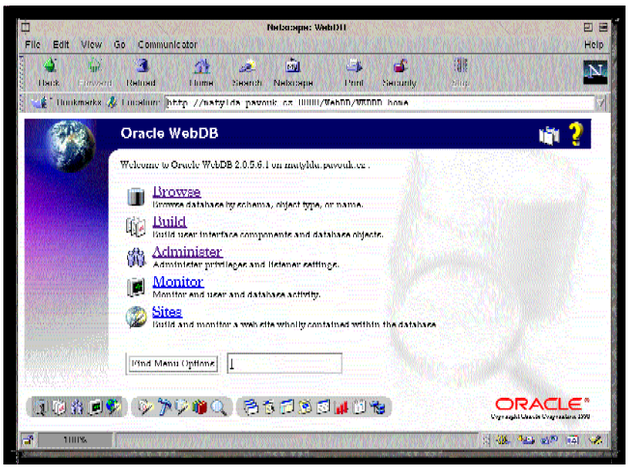
Kromě WinNT a většiny Unixů nás bude zajímat verze pro Linux, kterou jsme také zakoupili a instalovali. Dále se budeme věnovat právě verzi pro náš oblíbený Linux.



Instalace celého balíku

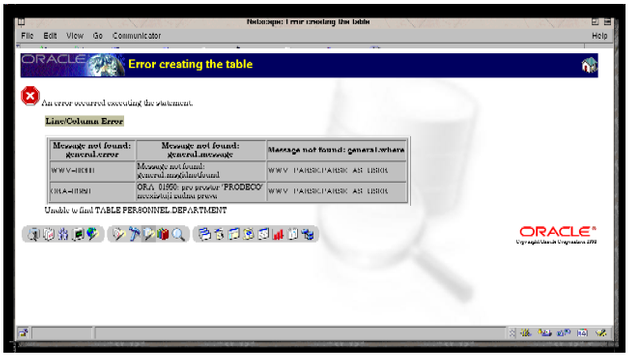
V poměrně dobře připravených instalačních příručkách je nejdůležitější požadavek na glibc 2.0.7 a jádro 2.0.34 nebo vyšší v podstatě jediný v případě databáze. U disků instalační příručka doporučuje použít více malých disků než jeden velký. Je to otázka bezpečnosti a záleží jen na uživateli, jak moc si cení svých dat. Budete-li chtít tomuto požadavku vyhovět, vězte, že budete potřebovat 4 mount pointy. Bezpečnost systému můžete však zajistit i jinak – např. RAIDem. Ostatní minimální HW požadavky opravdu nejsou nic moc – 32MB RAM, trojnásobek swapu, 400MB HDD pro plnou instalaci a samozřejmě mechanika CD. Jak dobře však DB poběží na takovémto stroji si nedovedu představit.

Instalační příručka WebDB už požaduje přímo RedHat 5.1 a nainstalovaný databázový motor 7.3.4, 8.0.5 nebo 8i.



Na čem jsme pracovali

Jako databázový server byl použit PC od Vikomtu (model Danaus) s procesorem PIII, jedním diskem 13GB a 384MB RAM. Instalovali jsme Debian 2.1 s jádrem 2.2.11 (ovšem rychle nahrazeným 2.2.12) a přistoupili k instalaci DB.

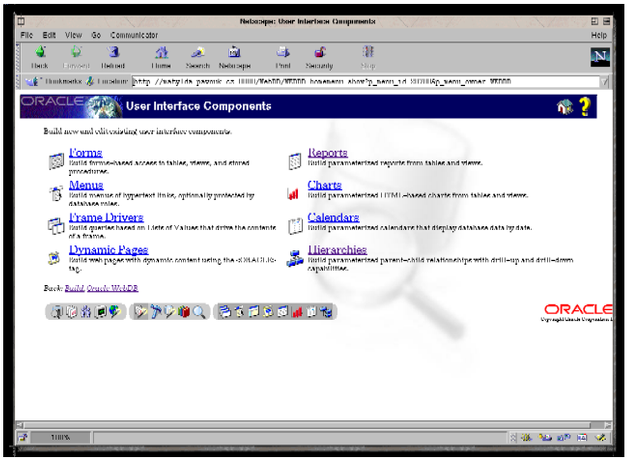


Poznámky k instalační proceduře DB

Nejprve databáze – znalci nechť tento odstavec přeskočí :)

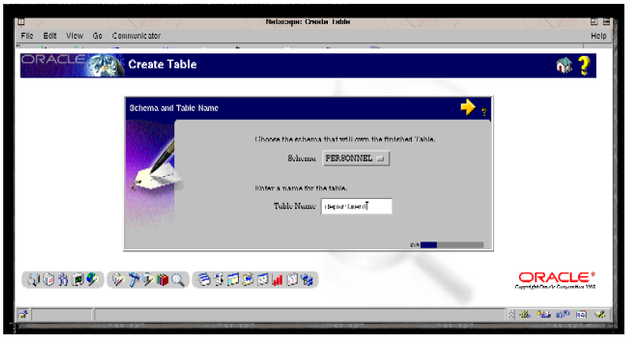
Po vytvoření mountpointů a uživatele (majitele databáze) je potřeba spustit skript oratab.sh. Příručka Vás neupozorní, ale skript ano – před spuštěním je potřeba nastavit proměnnou prostředí ORACLE_OWNER. Navíc skript požaduje sh v /usr/bin, který jsem zatím viděl jen v /bin. Takže jsme skript zkopírovali na disk, opravili první řádku a nahradili používanou proměnnou GROUPS na GROUPSS (jinak sh hlásil, že tato proměnná je jen pro čtení, což je zřejmě problém bashe 2.0, na RedHatu mu to nevadilo). Pokud si při instalaci zvolíte instalaci i dokumentace k RDBMS, asi Vás zaskočí hláška o chybějícím adresáři. Pomůže ruční zásah z jiného terminálu. Také si předem zkontrolujte, máte-li /usr/lib/libtcl.so. Máte-li nainstalovánu jen jednu verzi tcl, asi to bude v pořádku. My jsme měli tcl 7.6 i tcl 8.0 a museli jsme ručně vytvořit link na příslušnou knihovnu (verze 8.0). Tento problém se také objevuje pouze u Debianu.

Po instalaci se musí spustit skript root.sh (jak požaduje instalátor i instalační příručka) – opět záležitost s požadavkem na /usr/bin/sh. Na závěr je potřeba ještě spustit `catrep.sql` pomocí sqlplus (takže jej nezapomeňte nainstalovat). Provede se to příkazem: `@upl-na_cesta_ke_scriptu`. Při běhu tohoto skriptu proběhlo velmi mnoho hlášek, mezi nimi i nějaké chybové. Alespoň si to myslíme, ale bylo to moc rychlé a v logu jsme nic nenašli. Podle pozdějšího hovoru s hot-line je to v pořádku. Skript si totiž před vytvořením některých objektů udělá (pro jistotu) jejich zrušení, no a když neexistují… Na závěr ještě musíte opsat spouštěcí skript do /etc/init.d (na RedHatu do /etc/rc.d/init.d). Také by mohl být na instalačním CD. Tím byla základní instalace Oracle databáze hotová.



A nyní to hlavní – instalace WebDB

Vždycky jsem si myslel, že příkaz `mount /dev/cosi /mnt/něco -t fs` udělá úplně stejnou věc jako příslušný zápis v `/etc/fstab` a zápis `mount /mnt/něco`. Až při instalaci WebDB jsem zjistil, že tomu tak není. Přimontování instalačního CD ROMu druhou variantou vede při spuštění ke hlášce „permission denied“. První varianta ovšem vede k cíli – instalační skript se spustí a funguje (a instalační příručka požaduje připojení CD právě takto). Instalační příručka obsahuje i obrázky obrazovek instalátoru (na rozdíl od instalační příručky DB). Zde už na uživatele nečeká žádný zádrhel a vše běží k plné spokojenosti. Jen v jedné chvíli to vypadá, že skript umřel – disk nevrčí, CD mechanika nebliká – vydržte, za chvíli to bude pokračovat. Na konci opět opíšete spouštěcí skript (tentokrát je to jen jeden řádek) a můžeme do testování. Doufám, že jste si všimli hlášení o uživatelském jménu a heslu v průběhu instalace.

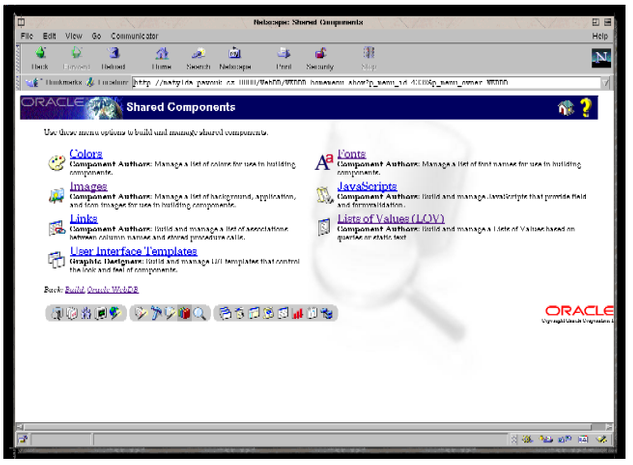


První spuštění

Nejdříve musíme samozřejmě správně spustit DB i WebDB

listener (o tom později). Pozor: listener spusťte jako root. Po spuštění brousku, zadání URL a autentifikaci se nám objeví úvodní obrazovka – obr 1. Máme zde základní volby další práce – prohlížení DB objektů, komponent atd; další možností je jejich tvorba, administrace (správa uživatelů, oprávnění, konfigurace listeneru,...) a poslední možností je monitorování systému.

Předpokládám, že každý začne vytvořením uživatele (sebe) a pod tímto uživatelem bude dále pracovat. Zde ovšem narazí na (již zdokumentovanou) chybu aplikace. Při pokusu o vytvoření tabulky (očekávám, že to bude druhým krokem) projde celým průvodcem až ke konečné chybové hlášce o nenalezených právech pro tabulkový prostor – viz obr. 2 (ano, DB mluví česky). Náprava je jednoduchá – každému uživateli, který má být vývojářem (role webdb_developer) je potřeba přidat ještě oprávnění „resource“. Pak bude vše probíhat dobře (tedy pokud si správně nastavíte práva). HotLine (zaslouží si opravdu pochvalu) ještě poradila po instalaci spustit skript catalog.sql (kromě již uvedeného catrep.sql) – o něm se instalační příručka nezmiňuje.



Nabodeníčka

Čeština je v současné době neúplná. Samotná databáze česky umí (to však asi poznáte jen z chybových hlášek), ale nyní dodávaná werze WebDB (2.0.5) češtinu ve svých možnostech nemá – tedy nemá česká menu. Jinak češtinu v datech samozřejmě používat můžete. ALE během velmi krátké doby (v řádu dní) bude dodávána nová verze a ta již česky umět má. A způsob nastavení jazyka je hezky řešen. Podle nastavení jazyka prohlížeče se automaticky nastaví jazyk odpovědí. Tedy systému WebDB samozřejmě. Aplikace a její jazyk je v rukou jejího tvůrce.

Ovšem aby vše česky běhalo, musíte vše správně nastavit. Tzn. nastavit vlastnosti DB a správně spustit WebDB listener. Abych Vás nenapínal a ušetřil Vám hledání v dokumentaci, tedy věžte, že stačí v souboru init<sid>.ora přidat dvě proměnné. No tři, jak uvidíme. Takže na konec souboru přidáme nejdříve dva řádky:

```
nls_territory="Czech Republic"

nls_language="CZECH"
```

Všechny proměnné týkající se národního prostředí začínají `nls_`. Stačí nastavit jenom tyto dvě proměnné, ostatní se nastaví podle nich a nastavit je musíme jen v případě, že je chceme změnit. Do spouštěcího skriptu DB ještě můžeme pro jistotu přidat řádek (a do skriptu pro WebDB listener POVINNĚ) přidáme:

```
export NLS_LANG=\
        "CZECH_Czech Republic.EE8ISO8859P2"
```

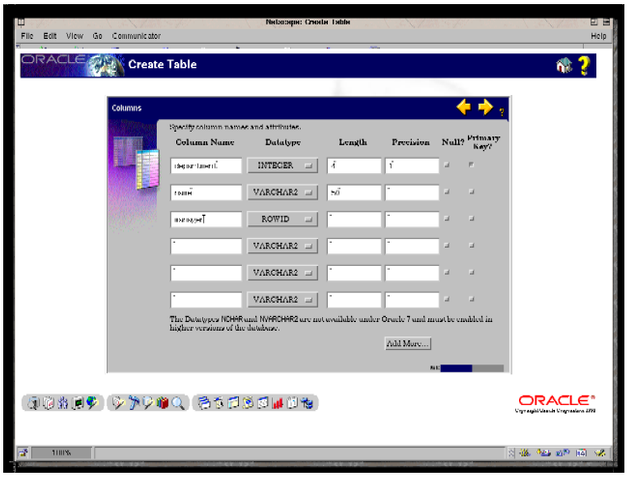
Když nyní spustíme DB a v sqlplus zadáme dotaz `select parameter, value from nls_session_parameters`, měli bychom obdžet takovouto odpověď:

PARAMETER	VALUE
NLS_LANGUAGE	CZECH
NLS_TERRITORY	CZECHREPUBLIC
NLS_CURRENCY	Kč
NLS_ISO_CURRENCY	CZECHREPUBLIC
NLS_NUMERIC_CHARACTERS	,.
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	DD.MM.YY
NLS_DATE_LANGUAGE	CZECH
NLS_SORT	CZECH

9 řádek vybráno.

No a podle výpisu vidíme, že asi budeme chtít nastavit NLS_DATE_FORMAT na „DD.MM.YYYY“ – přece jen, tisíciletí se blíží. Takže doplníme init<sid>.ora o tuto proměnnou. To se však týká standardního výpisu datových atributů. V reportech a pod. si můžete formát data taktéž specifikovat, takže je to jen na Vašem zvážení.

Abychom měli jistotu, můžeme si ve WebDB vytvořit report založený na uvedeném dotazu a měli bychom obdržet stejný výsledek. Jestliže ano, jsme za vodou...



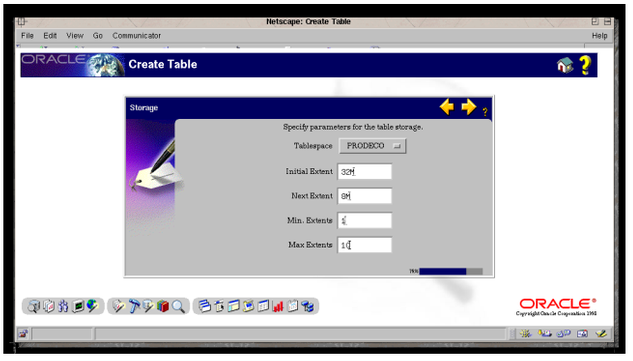
Neodpustím si malou poznámku: v kapitole o národním prostředí dokumentace (v angličtině) se hovoří o tom, že máme místo typů CHAR, VARCHAR a VARCHAR2 používat NCHAR nebo NVARACHAR2. Zkusil jsem to a při pokusu o vložení dat jsem získal chybovou hlášku o tom, že si neodpovídají znakové sady. Použil jsem tedy VARCHAR2 a vše fungovalo OK. Asi budu muset ještě pilovat angličtinu...

K čemu WebDB použijeme

Předpokládám, že ti, kdo budou mít zájem, se obrátí na (1), proto jen stručně. Celá tato sestava slouží k vytváření

int{er|ra}netových databázových aplikací. A výhoda oproti sestavě Apache/MySQL? V první řadě schopnosti databáze Oracle (bez komentáře) a navíc celá aplikace je kompletně uložená v databázi. Tedy konec změti statických stránek a CGI skriptů. I když nic nebrání jejich dalšímu používání a vybudování trojkombinace – Apache/WebDB/Oracle a koneckonců i MySQL a perl/PHP, skripty mohou dále plnit své úkoly. Nezanedbatelnou výhodou je také způsob „programování“ – vše probíhá v prostředí brousku formou průvodců. Má to ale háček, průvodci Vám neumožní 100% využít možností databáze, ale v záloze je textový klient sqlplus, takže nic není ztraceno. A pokud chcete ručně programovat – i to můžete.

Obrovským kladem produktu je i množství průvodců pro mnoho různých ovládacích prvků, reportů, grafů a dalších a dalších – viz obr. 3 a 4. Prakticky vše, co byste mohli potřebovat při vývoji www aplikace, tu najdete. Nepočítejte však s tím, že uděláte aplikaci na úrovni SAP R/3. Ovšem takové ty evidence, které má každá firma (kniha odeslané pošty, evidence čehokoliv apod), jsou „udělatelné“ během velmi krátké doby (začnete-li jako úplní laici, zvládnete to do týdne – asi).



První postřehy

Budete-li vytvářet tabulky, používejte průvodců (viz obrázky 5 až 8). Je vidět, že asi nevyužijete všech možností – např. klausule DEFAULT a další. Pro jednoduché tabulky to nevadí. A budete-li tvořit nějakou složitější datovou strukturu, použijete stejně asi nějaký CASE nástroj nebo si po analýze napíšete ručně vytvářecí skript (ještě že máme `vim`, že?) a použijete klienta sqlplus.

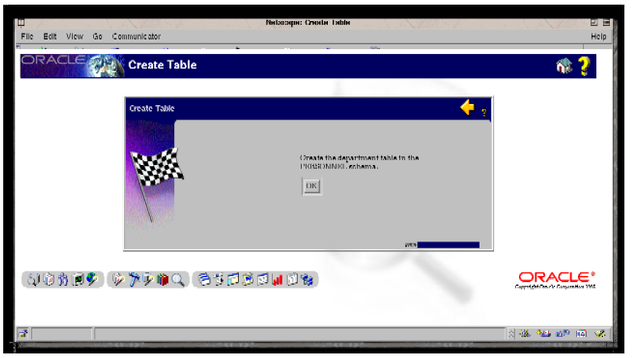
A když už něco vytvoříte, je občas potřeba něco zrušit (DROP…) nebo změnit (ALTER…). Tuto možnost jsem však zatím nenašel. A ani nenajdu (podle hot-line). Takže opět sqlplus.

Pokud Vás zajímá, jak je to s transakcemi, vězte, že WebDB transakce používá, ale provádí automaticky COMMIT po stisku tlačítka „uložit“. Pokud Vám toto nebude vyhovovat, budete muset trochu programovat. Zajistit se to však prý dá (nezkoušeli jsme, zatím…).

Co se týče textového klienta, musím říci, že obdobný klient pro MySQL se mi líbí víc (už jenom pro historii příkazů).

Naše dosavadní dojmy

Kromě uvedených poznámek (jsou opravdu všechny zatím objevené) jsme s produktem spokojeni, ba přímo nadšeni. Sice jej máme teprve několik dní, ale stojí to za to. Na uvedeném stroji je rychlost odezvy uspokojující – ovšem uvidíme, až server zatížíme plným provozem. Po grafické stránce se jedná o velmi povedený produkt. A pokud budete při návrhu prvků, reportů a stránek používat defaultní grafické rozvržení, určitě se nebudete muset za své výtvory stydět. A možností, jak si svá díla nastavit je ale opravdu mnoho, takže Vaše hotové aplikace se nemusí ani trochu podobat defaultnímu provedení.



Závěrem

Pokud Vás produkt zaujal, doporučuji jej Vašemu zájmu. Podle informací fy. Oracle se připravuje kniha o WebDB a má obsahovat i CD s instalací WebDB pro Linux. ■

1 Oracle www
`www.oracle.cz`

Solid

Jakub Dadák

Databázový server Solid vyvinula finská firma Solidtech a nyní je k dispozici ve verzích 2.3 a 3.0. Solid je SQL server, který nabízí robustní transakční zpracování dat od malých databází až po velké databáze rozmístěné na více mediích. Je velmi nenáročný na administraci a pracuje na většině běžných platforem. Podporuje množství komunikačních protokolů pro komunikaci klienta se serverem, klientský přístup přes ODBC (CLI API), JDBC a je navržen v souladu se standardy ANSI SQL2. Pro uživatele Linuxu je zajímavý tím, že jako jeden z prvních zahrnoval Linux do podporovaných platforem již dlouho před ostatními firmami a jejich RDBMS (Oracle, Informix, ADABAS,…) a umožnil tak na Linuxu vyvíjet komerční aplikace s robustní databázovou podporou, která do té doby byla jen velmi slabá (Postgress, MySQL atd.).

Solid může být provozován jak na běžném notebooku, tak na velmi výkonných víceprocesorových strojích. Multithreadová architektura, podpora uložených procedur a „row-level“ transakce z něj dělají databázový stroj použitelný i pro velmi náročné aplikace.

Databázový stroj

Vývojáři Solidu si kladli za cíl vyvinout robustní databázi s velmi malými nároky na systémové zdroje se snahou o jejich maximální využití. Stroj zpracovává dotazy přes Solid CLI API, poté jsou zpracovány SQL parserem a optimize-

Operating System Version	Supported network protocols	
AIX 3.2	TCP/IP, UNIX Pipes	ST
ChorusOS Classix Intel 3.1	TCP/IP	MT
Digital UNIX 4.0	TCP/IP, UNIX Pipes	MT
FreeBSD 2.2.2	TCP/IP, UNIX Pipes	ST
HP-UX 11.x	TCP/IP, UNIX Pipes	MT
Irix 6.3	TCP/IP, UNIX Pipes	MT
Linux Intel (glibc2) 2.0.34	TCP/IP, UNIX Pipes	MT
Linux Intel (libc5) 2.0.34	TCP/IP, UNIX Pipes	ST
Netware 4 4.1	IPX/SPX	ST
Netware 5 5.0	TCP/IP, IPX/SPX	ST
OpenVMS Alpha 6.1	TCP/IP, DECnet	ST
Sun Solaris Sparc 2.5, 2.6	TCP/IP, UNIX Pipes	MT
UnixWare 2.1	TCP/IP, UNIX Pipes	ST
VxWorks 5.3	TCP/IP	ST
Windows NT Intel 4.0	DECnet Client, IPX/SPX Client, Named Pipes, NetBIOS, Shared Memory, TCP/IP	MT
Windows 95/98 4.0, 4.10	DECnet Client, IPX/SPX Client, Named Pipes, NetBIOS, Shared Memory, TCP/IP	MT
Windows NT Alpha 4.0	DECnet Client, IPX/SPX Client, Named Pipes, NetBIOS, Shared Memory, TCP/IP	MT
Default: Client and Server available; Client = Client available Threads MT=Multithreaded ST=Singlethreaded		

Výpis č. 2: Tabulka hardware Solid

rem. Databázový stroj získává a ukládá data do databázových souborů. Databázový stroj podporuje:

- SMP multi-threadovou architekturu a paralelní zpracování
- Inteligentní row-level zamykání
- Unikátní kombinaci pesimistického a optimistického zamykání
- „Multiversioning“ pro zajištění konzistentního pohledu na data bez potřeby zamykání pro klientské procesy
- Podpora textových sloupců
- Komprimované indexy
- Automatické „roll-forward“ zotavení
- Škálovatelnost od notebooku po SMP RISC
- Minimální požadavky na operační paměť (min 300 kb)
- Binární kompatibilitu databáze na všech podporovaných platformách

Solid server využívá unikátní technologii „Bonsai Tree“. Jedná se o indexovací technologii, která uchovává informace o aktuálně prováděných změnách dat. Index umožňuje velmi rychle detekovat konflikty transakcí a snižuje tak nároky na ověření validity transakcí. Server podporuje dávkové inserty, které opět snižují vytížení harddisku. Pro permanentní uložení dat je použita varianta B stromu jak pro primární, tak pro cizí (foreign) klíče. Všechna data jsou ukládána do databázového souboru, který je mezi checkpointy (promítnutí transakčního logu do databáze) pouze pro čtení. Informace o probíhajících a potvrzených transakcích se ukládají do log souborů. Databáze může být tvořena více soubory na různých mediích. Maximální velikost jednoho atributu (sloupce) jsou 2GB a velikost databáze je omezena 32 TB. Server umožňuje explicitně nastavovat optimistické nebo pesimistické zamykání na úrovni tabulek a poskytuje celou řadu možností pro izolaci transakcí (read uncommitted, read commited, repeatable read, serializable).

Aplikační rozhraní

Klientské aplikace používají pro komunikaci se serverem jazyk SQL založený na syntaxi SQL89 Level 2 a ANSI SQL2 rozšíření.

SOLID SQL API je založeno na SQL Access Group’s call level interface standardu (SAG CLI), specifikaci pro dynamický SQL přístup k databázím. Solid CLI API je kompatibilní s ODBC specifikací, což činí databázi velmi dobře přístupnou win32 vývojářským systémům, které stále na Linuxu zatím k dispozici nejsou a velmi tak usnadňují vývoj,

který může probíhat na win32 a je pak zcela bezproblémově portován na Unix (Linux). K dispozici je též JDBC rozhraní pro přístup k databázi z Javy. CLI API je k dispozici ve statické i dynamické verzi C knihovny. Nad nativním API solidu byla napsána celá řada dalších rozhraní, např. pro Perl či PHP.

Instalace

Instalace serveru je velmi jednoduchá a zvládne ji i začátečník. Celý proces instalace je popsán v několikakilobytovém souboru. Instalační balík ve formátu tgz se rozbalí do zvoleného adresáře a instalační program pouze vyžaduje zadat jméno a heslo pro administrátora a licenční klíč. Automaticky je vytvořena prázdná databáze podle defaultního konfiguračního souboru, která naprosto dostačuje pro běžné aplikace nebo testování. Tím instalace končí, narozdíl od několikahodinové práce např. u Oracle či Informixu, kdy se zvláště začátečník prokousává tunami často zavádějící dokumentace (u Linuxových portací je to zvykem). Firma Solidtech umožňuje získat jejich databázový server pro testovací účely přímo z jejich www serveru (1). Jedná se o plně funkční verzi omezenou pouze platností 30-ti denní testovací licence.

Správa

Správa serveru je stejně jednoduchá jako instalace. Administrátor se de facto stará pouze o zálohování záloh databáze, které se vytvářejí automaticky podle konfigurace do speciálního adresáře. Zálohuje se vždy konfigurační soubor, soubor(y) databáze a soubor(y) transakčního logu. Obnova poškozené databáze znamená tedy pouze obnovu těchto souborů. Správa databáze je prováděna aplikací solcon (Solid Console), která umožňuje kromě záloh a checkpointů sledovat stav serveru, zapnout monitoring, sledovat uživatele a restartovat server. Samotný server je pouze jeden binární soubor, jehož spuštěním s parametrem adresáře databáze se startuje i celý databázový systém.

Podporované platformy a OS

Solid v současné době podporuje značné množství hardwarových platforem a OS. Přehledně to znázorňuje Tabulka hardware Solid převzatá z WWW firmy Solidtech.

Zkušenosti s provozem pod OS Linux

Solid v naší firmě provozujeme zhruba 2 roky a máme s ním velmi dobré zkušenosti. Server je velmi stabilní i při velké zátěži a i na databázích kolem 1GB vykazuje při poměrně složitých dotazech slušné výsledky, srovnatelné např. s MS SQL 6.5. Je pravda, že za Oraclem při větších objemech dat poněkud zaostává. Podle mého názoru se Solid nejlépe uplatní jako databázový server pro WWW servery a středně velké aplikace, které vyžadují bezpečné a spolehlivé uložení dat. Mezi hlavní výhody řadím i jeho nenáročnost na správu a rychlost nasazení. Ještě donedávna byly na WWW serveru firmy Solidtech uvedeny ceny (pro verzi 2.x), od verze 3.0 se však změnily i stránky a ceny už jsem nenašel. Obchodníci ze Solidu mi však sdělili, že cena Solidu pro 25 spojení a 1 CPU činí pro Linux a WinNT 1995$, pro ostatní platformy je to 3295$. Solid je ochoten udělat cenu i na míru v závislosti na aplikaci a počtu zakoupených licencí. Pokud jste se tedy ještě nerozhodli jaký SQL server na Linuxu používat, tak rozhodně Solid vyzkoušejte, určitě Vás nezklame. ■

1 Solid
http://www.solidtech.com

Oracle pro Linux

Radim Kubacki

Oracle je v současné době dostupný pro Linux v následujících verzích: Oracle 8.0.5(.1) Standard a Enterprise Edition a ve verzi Oracle 8i označovaná také jako 8.1.5 opět jako Standard i Enterprise Edition. Mimo to je zde také Oracle Application Server ve verzích 4.0.7 a 3.0.2 a produkt WebDB. Vše je možné je nalézt na adrese (1), příp. je možné si jej stáhnout pomocí ftp ze serveru (2).

Oracle je jednou z nejrozšířenějších databazí. Verze SE a EE jsou víceuživatelské, transakční relační databázové servery. Je možné škálovat jejich výkon a použít je i v opravdu rozsáhlých aplikacích. Mimo Linuxové verze existují i její ekvivalenty na většině Unixových systémů (Solaris, HP-UX, AIX,...), VMS, Windows NT. V rámci Enterprise Edition verze jsou dostupné navíc oproti jednodušší verzi tyto rysy: Advanced Replication, Object a Partitioning Option, Spatial, Time Series a Visual Information Retrieval Cartridge, Parallel Query a Advanced Networking Option. Firma Oracle přitom pro Linuxovou verzi poskytuje zdarma vývojářskou licenci jednomu vývojáři na jeden stroj. Licence zdarma se tedy netýká produkčního použití této databáze.

Dále se zaměřím především na verzi 8.0.5, protože Oracle 8i má prozatím ve verzi pro Linux příliš mnoho chyb. Navíc došlo při přechodu od verze 8.0.5 k 8i k přepsání instalátoru do Javy a budete tedy potřebovat Java Runtime Environment, X Window System a především dostatek paměti k samotné instalaci. 128 MB doporučených výrobcem je údajně opravdu potřebných. Poté, co přežijete instalaci, už nároky budou vcelku srovnatelné s 8.0.5. Nicméně bývá doporučováno v případě zájmu o tento produkt počkat na novou verzi, která je ohlášena do konce tohoto roku. Hlavní rozdíl oproti řadě 8.0 je přitom optimalizace pro internetovské aplikace, a to v rozsahu, který je větší než by naznačovala změna v číselném označení produktu.

Instalace

Po stažení potřebných souborů nebo jejich získání na instalačním CD (soubory budou zabírat 180 až necelých 300MB podle zvolené verze) můžete přikročit k instalaci.

Verze 8.0.5 je původně certifikována s distribucí RedHat 5.2, ale je možné ji provozovat i na strojích s RedHat 6.0 nebou jinou srovnatelnou konfigurací. V takovém případě se ovšem instalace poněkud zkomplikuje, protože je nutné aplikovat patche, které zajistí kompatibilitu s potřebnými knihovnami opětovným slinkováním programů (Oracle 8.0.5 je linkován s glibc2.0). Tyto patche naštěstí získáte na stejném místě jako samotný Oracle. Průběh instalace této verze je poměrně podrobně popsán na (3). Dozvíte se tam, v jakém pořadí provádět jednotlivé instalační kroky včetně úpravy některých parametrů jádra a jak předejít některým chybám, na které můžete narazit. Celkově budete potřebovat aspoň 400MB diskového prostoru a stroj s minimálně 32MB paměti.

Závěrečným krokem takové instalace je vytvoření nové databáze. Pokud tak neučiníte přímo v instalátoru, budete to muset udělat přímo. Potřebujete si k tomu připravit vlastní init<sid>.ora soubor, který získáte úpravou existujícího souboru obsaženého v distribuci. <sid> je označení jedné konkrétní instance databáze. Pak si nezapomeňte nastavit proměnnou prostředí ORACLE_SID na zvolenou hodnotu <sid> a můžete spustit Server Manager (svrmgrl). V tom můžete založit vlastní databázi, výhodné je použít skriptů crdb.orc a crdb2.orc, které vhodně upravíte. Naleznete je po instalaci v adresáři $ORACLE_HOME/rdbms/install/rdbms. Tyto skripty vytvoří nejen samotnou databázi, ale přidají i založení rollback segmentů, můžete nařídit i založení tablespaců a zajistí vygenerování tzv. data dictionary. Automaticky jsou vytvořeni dva uživatelé SYS a SYSTEM, kterým je přidělena role databázových administrátorů. Hesla těchto uživatelů naleznete v dokumentaci, když nahlédnete do Oracle Administrator's Guide.

S takto instalovanou databází můžete komunikovat mnoha způsoby. Nejjednodušeji je to možné pomocí produktu SQL Plus, který vám umožní zadávat příkazy jazyka SQL a sledovat jejich výsledky. Stojí za to připomenout, že je podporováno SQL92 Entry Level a řada rozšíření definovaných firmou Oracle. Jedním z nejvýraznějších je existence procedurálního ijazyka PL/SQL dovolujícího psaní procedur a funkcí. Pro programování můžete použít Pro*C, což je prekompilátor, který vám umožní používat v C/C++ programech SQL příkazy jazyka SQL použitím direktiv EXEC SQL <sql_command>. Ty jsou v průběhu prekompilace nahrazeny voláním funkcí C jazyka v dodaných knihovnách. Tyto knihovny můžete samozřejmě používat také přímo, dokumentace pro Oracle Call Interface (OCI) je součástí produktu. Pracujete-li s jazykem Java, můžete použít dostupné JDBC drivery, a to jak ovladače v čisté Javě, tak i verzi používající OCI knihovny. Existují i moduly rozšiřující jiné jazyky, jako je např. Perl, PHP.

Jak funguje Oracle

Oracle na vašem počítači vytvoří mnoho souborů. Část z nich budou programové soubory, které získáte při instalaci. Dále ke každé existující databází existuje aspoň jeden control file popisující konfiguraci dané databáze. Je lépe mít víc kopií tohoto souboru, nejlépe na různých discích pro případ poškození. Data každé databáze, v terminologii Oraclu nazývané instance, se ukládá do jednoho nebo více tablespaců. Fyzicky je každý tablespace reprezentován datovými soubory. Zároveň je tablespace důležitý nástroj administrace a správy databáze. Ukládají se do nich tabulky, indexy, package, procedury, pohledy (views), synonyma a nově vytvořené typy. Vhodnou definicí vlastností tablespaců, která zahrnuje jejich fyzické umístění, pravidla pro přidělování prostoru na disku, kvóty uživatelů, a správně zvoleným rozmístěním objektů do nich lze dosáhnou optimalizace výkonu databáze.

Další věcí nezbytnou pro chod Oraclu jsou rollback segmenty, do kterých se ukládají historie prováděných transakcí, musí být aspoň dva pro každou instanci a jejich počet se odvozuje od počtu současně prováděných transakcí. Opět jsou umístěny do zvolených tablespaců.

Na rozdíl od verze Oracle7 přichází verze 8 s tzv. Oracle Flexible Architecture, což je množství doporučení, jak pojmenovávat a kam ukládat jednotlivé soubory. Není však nutné se jich bezvýhradně držet.

Databázový server při spuštění každé instance vytvoří několik procesů, které jsou pojmenovány ora_????_<sid> a společně zajišťují jeho činnost. Navíc můžete potřebovat listener pro podporu komunikace prostřednictvím TCP/IP protokolu. Každé připojení klienta k databázi pak má za následek vytvoření dalšího procesu. Ladiči výkonu mají velké množství možností nastavování parametrů ovlivňujících činnost databáze. Systém je celkově velice stabilní a nedochází v něm ke ztrátám dat. Pokud se tedy vyhnete nekorektnímu ukončování běžící databáze a nepoškodíte či nesmažete si některý ze souborů sami, nebudete zřejmě mít problémy. Komu však záleží na datech, bude zálohovat. Lze k tomu opět využít možnosti produktu prověřené dlouhým vývojem.

Národní prostředí

Potěšující zpráva říká, že Oracle podporuje češtinu. Není problém s použitím znakové sady ISO 8859-2, tříděním, jsou definovány i formáty pro datum, měnu. Navíc jsou počeštěny i některé katalogy zpráv. Pokud nezapomenete zadat správnou kódovou stránku (EE8ISO8859P2) už do skriptu pro vytvoření databáze a nastavení nls_language a nls_territory do init.ora souboru, je prakticky vystaráno. Důležité je zejména říci hned při zakládání databáze, že chcete tuto sadu. Nastavení NLS (National Language Support) proměnných je možné měnit pro každou session zvlášť.

Další vývoj

Zájem o Linux ze strany firmy Oracle nekončí uvedením jmenovaných produktů. Přislíbeny jsou nové verze jak samotné databáze, tak i aplikačního serveru. Navíc se ještě letos připravuje uvolnění jednodušší verze databázového stroje Oracle 8i Lite, a dokonce má být portováno i vývojové prostředí Oracle Developer. ■

1 Oracle
http://technet.oracle.com/
2 Oracle ftp
ftp://ftp.oracle.com/
3 Jordan
http://jordan.fortwayne.com/oracle

PostgreSQL SPI a triggery v C

Karel Žák

Doslovný překlad slova trigger je spoušť a nebo ještě lépe spouštěč. To přesně vystihuje činnost, kterou trigger vykonává. Jedná se o uživatelem definovanou funkci, která je spuštěna nad předem definovanou tabulkou v předem definované situaci (např. při použití INSERT, DELETE, UPDATE). Uživatel tak do rukou dostává velmi silný nástroj. Typické použití triggeru muže být například při hlídání vzájemné konzistence několika tabulek, úpravě dat před jejich uložením apod.

V PostgreSQL (PgSQL) se trigger vytvoří přikazem CREATE TRIGGER. Tímto SQL příkazem se přiřadí již dříve definovaná funkce k určité tabulce a určité akci nad touto

```
#include "executor/spi.h"
#include "commands/trigger.h"
#include "miscadmin.h"

HeapTuple       nowin(void);

HeapTuple nowin()
{
        TupleDesc       tupdesc; /* data description */
        HeapTuple       rettuple = NULL; /* tuple data */
char *data, /* my data */
*tablename, /* table name */
*action = NULL, /* SQL action - INSERT..*/
Qbuff[8*1024]; /* buffer for SQL query */
int attnum, /* column number        */
ret;

        if (!CurrentTriggerData)
                elog(ERROR, "triggers are not initialized");

        if (TRIGGER_FIRED_BY_UPDATE(CurrentTriggerData->tg_event)) {
                action = "UPDATE";
                rettuple = CurrentTriggerData->tg_newtuple;
        } else if (TRIGGER_FIRED_BY_INSERT(CurrentTriggerData->tg_event)) {
                action = "INSERT";
                rettuple = CurrentTriggerData->tg_trigtuple;
        } else if (TRIGGER_FIRED_BY_DELETE(CurrentTriggerData->tg_event)) {
                action = "DELETE";
                rettuple = CurrentTriggerData->tg_trigtuple;
        }
        tupdesc = CurrentTriggerData->tg_relation->rd_att;

        /* Connect to SPI manager */
        if ((ret = SPI_connect()) < 0)
                elog(ERROR, "SPI_connect returned %d", ret);

/* Set table name */
tablename = SPI_getrelname(CurrentTriggerData->tg_relation);

/* Log action */
        sprintf(Qbuff, "INSERT INTO logOS VALUES ('%s', '%s', '%s', 'now'::text)",
tablename, GetPgUserName(), action);
        ret = SPI_exec(Qbuff, 1);
        if (ret < 0)
                elog(ERROR, "SPI_exec returned %d for LOG", ret);

elog(NOTICE, "Hello - here LN trigger (for %s on %s)!", action, tablename);
```

Výpis č. 3: Triggerová funkce část 1.

tabulkou. Jak již bylo zmíněno, je nutné, aby funkce byla definována ještě před použitím příkazu CREATE TRIGGER. To se provede příkazem CREATE FUNCTION, ve kterém se definuje jméno funkce, místo uložení jejího spustitelného kódu, programovací jazyk, ve kterém je funkce napsána a datový typ který funkce vrací (u triggerové funkce to musí být typ 'opaque'). V PgSQL je možné používat na psaní funkcí libovolný programovací jazyk (v PgSQL se takový jazyk nazývá Procedural Language – PL). V současné době je ve standardní distribuci podpora pro funkce napsané v PL/pgSQL, PL/Tcl, C a pracuje se na Perlu. Pokud někdo chce, může si do PgSQL přidat i vlastní podporu pro jiný jazyk (tedy pokud do něho implementuje operace potřebné pro práci se SPI rozhraním serveru). Nový jazyk se definuje příkazem CREATE LANGUAGE. Více informací naleznete v PostgreSQL Programmer's Guide, která je např. na (1).

V tomto článku se budeme věnovat psaní nových funkcí v klasickém C. V případě C se SQL serveru předává nová funkce v podobě .so souborů. Tedy klasického zdrojového kódu zkompilovaného např. v `gcc` s parametrem `-shared` (více viz ukázka). Jedná se tedy v podstatě o dynamické přidání nového modulu (podobně jako u kernelu nebo DSO-apache atd.) bez nutnosti nějak modifikovat a kompilovat SQL server.

Ještě dříve než se dostaneme k psaní vlastního triggeru, je nutné se zmínit o SPI (Server Programming Interface), které budeme v triggerové funkci používat.

```
/* Not action for DELETE */
if (!strcmp(action, "DELETE")) {
                SPI_finish();
                CurrentTriggerData = NULL;
                return(rettuple);
}

/* OSname data */
attnum = SPI_fnumber(tupdesc, "OSname");
data   = SPI_getvalue(rettuple, tupdesc, attnum);

/* okna -to-> trash */
if (!strcmp(data, "okna"))
elog(ERROR, "Sorry, but 'okna' is not goodOS!");

/* penguin -to-> Linux */
else if (!strcmp(data, "penguin")) {
Datum   newdt; /* Datum is spec. data type */

/* set new data */
newdt   = PointerGetDatum( textin( "Linux" ) );

/* modify returned tuple */
rettuple = SPI_modifytuple(CurrentTriggerData->tg_relation,
rettuple, 1, &attnum, &newdt, NULL);
}
        SPI_finish();
        CurrentTriggerData = NULL;
        return(rettuple);
}
```

Výpis č. 4: Triggerová funkce část 2.

SPI – Server Programming Interface

PostgreSQL SPI poskytuje programátorům velmi silný nástroj pro zacházení s daty uvnitř samotného SQL serveru. Je tedy možné i poměrně náročné akce provádět na straně serveru bez nutnosti implementovat tyto činnosti u klienta. To může být velmi výhodné, pokud například vytváříte několik různých rozhraní (v PHP, C, Perl...) a nechcete v kazdém z nich implementovat to samé.

SPI je v některých případech velmi podobné standardní klientské libpq. I zde je možné pokládat serveru SQL dotazy (ano, čtete dobře – uvnitř SQL serveru můžete používat SQL dotazy stejně jako u klienta), modifikovat data atd.

Základními funkcemi jsou `SPI_connect()` / `SPI_finish()`. Tyto funkce připojí/odpojí váš modul (vaši funkci) od/k SPI rozhraní serveru. Další důležitou funkcí je `SPI_exec()`, která umožňuje položit serveru SQL dotaz a definovat počet tuples, které mají být vráceny. Funkce vrací status odpovědi (je či není-li v pořádku). Zároveň nastavuje globální proměnné `SPI_processed` (počet odpovědí – počet tuples) a `*SPI_tuptable` (pointer na tuples – tedy na data).

V SPI programování je nutné rozlišovat u tuples dva důležité datové typy, a to:

- HeadTuple – ve `*SPI_tuptable` je to `SPI_tuptable->vals[n]`, kde 'n' je číslo řádky (tuple) odpovědi. Tato struktura obsahuje vlastní data.
- TupleDesc – ve `*SPI_tuptable` je to `SPI_tuptable->tupdesc` a obsahuje popis tuples (tuple description).

Tyto dvě struktury jsou základními stavebními kameny SPI i triggerů v C a jsou často parametrem SPI fukcí. Například pokud chceme vědět pořadové číslo sloupce (správně atributu) 'ahoj' v odpovědi, tedy použijeme:

`SPI_fnumber(SPI_tuptable->tupdesc, "ahoj");`

K datům v tuples se dostanete např. pomocí funkce `SPI_getvalue()`. Důležité je nezapomenut na to, že SQL server interně pro SPI nerozlišuje datové typy (tak jak jsou definovány v tabulkách (CREATE TABLE), ale v SPI je vše řetězec.

Např.:

```
int radka   = 10;
int sloupec = SPI_fnumber(\
              SPI_tuptable->tupdesc, "ahoj");
char *data  = SPI_getvalue(\
              SPI_tuptable->vals[radka],\
             SPI_tuptable->tupdesc, sloupec);
```

Tuples jde i modifikovat. To použijeme hlavně u triggerů. Modifikace je možná funkcí `SPI_modifytuple()`. Tato funkce přijmá data pouze v typu 'Datum' (což nemá nic společného s datem). Při konverzi do tohoto typu je např. typ 'char' nejdříve nutné přetypovat na typ 'text' a pak na typ 'Datum'. A to: `PointerGetDatum( textin( "neco" ))`. U 'int' je přetypování provedeno makrem `Int32GetDatum()`.

SPI obsahuje celkem 18 funkcí, jejichž přesný popis na-

jdete ve zmiňovaném programátorském manuálu. Pro účely tohoto článku vystačíme s již uvedenými.

C Trigger

Při definici triggeru příkazem CREATE TRIGER definujeme chvíli, kdy má být naše funkce (PROCEDURE) spuštěna před/po (BEFORE / AFTER) příkazy INSERT nebo DELETE nebo UPDATE a nad kterou tabulkou (ON <relation name>). Také je možné triggerovou funkci zavolat s parametry (args). Viz:

```
CREATE TRIGGER <trigger name> <BEFORE|AFTER>\
        <INSERT|DELETE|UPDATE>
        ON <relation name> FOR EACH <ROW|STATEMENT>
        EXECUTE PROCEDURE <procedure name>\
        (<function args>);
```

Ale vraťme se k programování. Při zavolaní naší triggerové funkce nám SQL server poskytne globální proměnnou (strukturu) CurrentTriggerData. Veškeré triggerové operace se týkají této struktury. Ta (mimo jiné) obsahuje:

- `CurrentTriggerData->tg_trigger` – obsahuje informace o triggeru (jeho jméno, args) atd.
- `CurrentTriggerData->tg_event` – proměnná, která nám umožní zjistit pomocí maker TRIGGER_FIRED_BY_UPDATE (INSERT, DELETE, při jakém SQL příkazu je trigger volán a kdy je volán TRIGGER_FIRED_AFTER (BEFORE).
- `CurrentTriggerData->tg_relation` – obsahuje informace o tabulce, na kterou je trigger volán.
- `CurrentTriggerData->tg_relation->rd_att` – je TupleDesc (viz. část o SPI) – tedy popis dat.
- Pro HeadTuple se v triggerech používá název 'rettuple'. Tou je `CurrentTriggerData->tg_trigtuple` u INSERT, DELETE (a obsahuje vkládaná nebo mazaná data). U UPDATE je to `CurrentTriggerData->tg_newtuple`, která obsahuje nová data, zatímco původní data jsou v `CurrentTriggerData->tg_trigtuple`.

Před vlastním příkladem snad ještě zmínka o funkci elog(typ, str). Ta, pokud použijete jako typ logu 'ERROR', ukončí transakci (a provádění triggeru), pokud použijete 'NOTICE', odešle 'str' klientovi (to může být výhodné například při ladění atd.).

Na příkladu Triggerová funkce část 1. si ukažme triggerovou funkci, která končí chybou, je-li do tabulky „goodOS“ a sloupce „OSname“ vkládán text "okna". Je-li do téhož sloupce vládán text „penguin“, je nahrazen textem „Linux“. Zároveň trigger uloží o každé akci log (tabname, username, action, datum) do tabulky „logOS“. Trigger bude spouštěn vždy pro každou řádku (FOR EACH ROW) pro INSERT, DELETE a UPDATE. To znamená, že v 'rettuple' budou data pro danou řádku.

Kompilaci tohoto příkladu provedeme příkazem:

```
gcc -shared -Wall -O3 -o LN.so LN.c
```

Pokud máte hlavičkové soubory někde jinde, nezapomeňte použít -I (např. já raději používám include soubory, které jsou ve zdrojovém balíku (poznámka: před jejich použitím je ale nutné nejdříve spustit `./configure`)).

Pak musíme SQL serveru sdělit, že máme novou funkci a chceme trigger, a je také nutné udělat tabulky, tedy:

```
DROP TABLE goodOS;
CREATE TABLE  goodOS (OSname varchar(32));

DROP TABLE logOS;
CREATE TABLE  logOS (
tabname name,
username name,
action  varchar(16),
datum datetime
);

DROP FUNCTION nowin();
CREATE FUNCTION         nowin()
        RETURNS         opaque
        AS              '<YOUR_FULL_PATH>/LN.so'
        LANGUAGE        'c';

DROP TRIGGER LN_trigger ON goodOS;
CREATE TRIGGER LN_trigger BEFORE INSERT\
        or DELETE or UPDATE
ON goodOS FOR EACH ROW
EXECUTE PROCEDURE nowin();
```

Na závěr malé upozornění. Naprogramováním triggeru v C přímo vstupujete do SQL serveru, uděláte-li tedy chybu ve své funkci (např. přístup do paměti který kernel odmění signálem SIGSEGV), ovlivníte tím chod SQL serveru a ten se může tedy odebrat do souboru `core`. Ale pochopitelně se jedná pouze o potomka hlavního SQL serveru (`postmastera`), není tedy nutné se bát, že takto ovlivníte celý server (to je také výhoda ne-threadového řešení serveru, i když za cenu větší pomalosti (viz. `fork()`).

Uvedený příklad (i zkompilovaný) naleznete na (2). ■

1 Docs Linux
http://docs.linux.cz
2 Příklad
http://home.zf.jcu.cz/~zakkr/LN/

DBI — Perlové rozhraní pro přístup k databázím

Jan Pazdziora, adelton@fi.muni.cz

Perl je jazyk pro zpracování (primárně) textů a je oblíbeným skriptovacím nástrojem WWW serverů, relační databázové servery jsou efektivním systémy pro uložení velkého množství dat a rychlý přístup k nim. Spojení Perlovské flexibility s databázovými možnostmi uchování velkých objemů údajů je pak častým úkolem i přirozeným řešením. Pro přístup k databázím slouží v Perlu DBI — DataBase Interface.

Typická relační databáze umí z uživatelského hlediska relativně málo věcí — provést SQL příkaz s nějakými parametry (INSERT, DELETE) či provést SQL dotaz a vrátit posloupnost záznamů (SELECT). DBI definuje jednotný způsob, jak databázi předat příkaz a jak dostat zpět výsledek. Protože představy různých databází o tom, jak se s nimi má pracovat, se velmi liší, je DBI navrženo modulárně. Základní DBI definuje pouze metody směrem k Perlovskému uživateli, pro konkrétní databázový server pak potřebujeme specifický DBD — DataBase Driver. Pro každý typ

databáze jiný driver. Výhodou takovéhoto modulárního rozdělení úkolů je velká nezávislost na straně Perlu, na druhé straně možnost implementovat práci s databází co nejefektivněji s využitím databázi na míru šitého driveru.

Instalace DBI i jednotlivých driverů probíhá standardně:

```
$ perl -MCPAN -e shell
cpan> install DBI
cpan> install DBD::mysql
```

či stažení, rozbalení a pak

```
perl Makefile.PL && make &&\
    make test && make install
```

Tím prvním příkladem naznačuji, že DBI i drivery jsou k dispozici na CPANu, tím druhým, že jsou to standardní moduly. Co už není tak standardní je nutnost číst při instalaci dokumentaci. Databázové drivery (DBD) totiž implementují rozhraní s databází a velmi často je nutno kompilovat driver s pomocí originálních databázových hlavičkových souborů a linkovat s originálními knihovnami, nezřídka za pomoci rozšířeného vývojového prostředí. Jinými slovy, před tím, než začnete driver kompilovat a instalovat, je dobré se přesvědčit, že máte opravdu zakoupeno/nainstalováno vše, co je potřeba. (Rostoucímu množství svých kolegů, kteří na svých systémech nemají překladač, ale přesto DBI chtějí, doporučujte použití ppm — ušetříte hodně času jak jim, tak sobě.)

Předpokládáme-li tedy, že jsme úspěšně nainstalovali jak DBI, tak DBD ovladač příslušný naší databázi, následuje první interakce s databázovým serverem — connect. Nejdříve je potřeba se k serveru připojit:

```
use DBI;
my $dbh = DBI->connect('dbi:mysql:test',\
    'user', 'pass',
    { RaiseError => 1, AutoCommit => 1 });
```

Zde ukazujeme, že pro připojení je nutno vědět, kam se vlastně připojujeme, a dále přihlašovací jméno do databáze a heslo. Za dbi: v prvním parametru connectu je jméno driveru, například mysql, Pg, Oracle, Informix. Za další dvojtečkou pak následuje bližší specifikace – typicky jméno databáze či databázové instance. Protože ale použité principy se server od serveru liší, je syntaxe této části přihlašovacího řetězce závislá na daném driveru — například prod, dbname=template1, host=www2:post=5553:db=test. Výsledkem connectu je tzv. databázový handler, reference na objekt, který ví, ke které databázi je připojen a jehož prostřednictvím můžeme serveru poslat SQL příkaz. Například

```
$dbh->do('delete from tabulka\
   where id = 123');
```

Většina databázových serverů podporuje (anebo je drivery emulují) bindované parametry:

```
$dbh->do('delete from tabulka\
   where id = ?', {}, $id);
```

Zde nespecifikujeme hodnotu přímo v SQL řetězci, ale předáváme ji jako parametr, běžný Perlovský skalár. Driver zajistí zpracování této hodnoty do podoby, kterou databáze je schopna zpracovat (případně provede expanci hodnoty do SQL příkazu, pokud databázový server bindované parametry sám o sobě nezná).

Toto jednorázové provedení příkazu je ve skutečnosti spojení několika akcí, které můžeme volat postupně:

```
my $sth = $dbh->prepare('insert into tab\
   values (?, ?, ?)');
while (<>) {
    chomp;
    my ($id, $login, $forward) = split /:/;
    $sth->execute($id, $login, $forward));
    }
```

Pomocí metody prepare (nad patřičným otevřeným database handlerem) nejprve připravíme tzv. statement handler s daným SQL příkazem, a pak opakovaně voláme provedení tohoto příkazu, pokaždé s jinými parametry, kdy hodnoty Perlovských skalárů jsou "dosazeny" na místa otazníků v SQL příkazu. Server tak SQL příkaz parsuje pouze jednou a při hromadném dávkovém zpracování ušetříme hodně času.

Jelikož jsme databázový handler otevřeli s atributem AutoCommit, je každý příkaz následován implicitním commitem. Pokud bychom atributem metody connect či přímo pomocí

```
$dbh->{'AutoCommit'} = 0;
```

autocommit mód vypnuli, provedeme pak explicitní ukončení transakce voláním

```
$dbh->commit;
```

(či $dbh->rollback). Podobně jako při předání SQL příkazu k provedení či zpracování bindovaných parametrů je věcí databázového driveru, aby Perlovské příkazy zaimplementoval tak, aby vyústily ve správnou akci databázového serveru.

Statement handler je možnou využít nejen k násobnému provedení jednoho SQL příkazu, ale také pro získání výsledků příkazu SELECT:

```
my $sth = $dbh->prepare('select name, age
            from people where dept = ?
                and salary > ?');
$sth->execute($oddeleni, $plat);
while (my ($jmeno, $vek) =\
    $sth->fetchrow_array) {
    $lide{$jmeno} = $vek;
    }
```

Zde je po provedení příkazu (execute) statement handler nachystaný na volání metod fetchrow_*, kterými postupně načteme jednotlivé záznamy. Opět, DBI ve spolupráci s konkrétním DBD zajistí, že hodnoty získané SELECTem jsou převedeny na rozumné Perlovské skaláry.

DBI poskytuje hodně metod a dokumentaci je rozhodně dobré přečíst – jednotlivé záznamy můžeme načíst jako hash indexovaný jmény sloupců, můžeme načíst celý seznam jedinou metodou, pomocí atributů jsme schopni vyladit chování daného spojení či říci, jak se má pracovat s položkami typu BLOB či CLOB, zda mají chyby vést přímo na die (RaiseError) či zda chceme, aby byla chybová hlášení pouze tištěna na standardní chybový výstup (PrintError) či chyby úplně ignorovány s tím, že budeme ošetřovat návratové hodnoty jednotlivých metod.

Většina toho je ale již pouze variací na zde předvedené téma: nedříve je potřeba provést connect do databáze, pak nad vytvořeným databázovým handlerem voláme metody, kterými zadáme SQL příkaz, případné parametry (buď po jednom pomocí bind_param, nebo jak jsme již viděli přímo jako další atributy do či execute), příkaz provedeme, a pokud je co číst, načteme (fetch*) výsledky.

Krása DBI spočívá v tom, že unifikuje metody napříč různými databázovými servery, tedy nemusíme si pamatovat, jaké parametry má mysql_connect ve srování s OCI8 — od toho nás odstíní patřičný databázový driver. Bindování parametrů pak nejen zvyšuje výkon odstraněním opakovaných nepotřebných volání, ale i pohodlí a bezpečí při programování, protože nad každou hodnotou převzatou z CGI formuláře nemusíme provádět ošetřování nepohodlých znaků, které se navíc opět server od serveru liší.

Relativně unikátním je databázový driver DBD::Proxy, který dovoluje vzdáleně přistupovat i k databázovým serverům, které jinak vzdálené připojení neposkytují, či ke kterým nemáme síťový modul zakoupený nebo kde klientské knihovny pro naši platformu neexistují. Pomocí proměnné prostředí DBI_AUTOPROXY či specifikací řetězce ve volání connect řekneme, ke kterému stroji a na jaký port se má náš skript připojit. Na vzdáleném počítači musí běžet DBI::ProxyServer, který spojení přijme a provede connect do lokální databáze. Všechny metody, které pak nad lokálním databázovým handlerem budeme volat, bude driver přeposílat na tento vzdálený ProxyServer a ten je předá databázovému serveru; zpět pak samozřejmě tečou výsledky.

Pokud programujete v Perlu, je DBI tím nástrojem, kterým přistupovat k databázím. Pokud potřebujete přistupovat k databázím a využít přitom výhodu skriptovacího jazyka s unifikovaným databázovým rozhraním (DBI je starší než specifikace ODBC), je Perl s DBI kandidátem, který je dobré brát v úvahu.

```
$dbh->disconnect;
__END__
```

■

Transakce v databázových serverech (a co s těmi, které transakce nemají)

Jan Pazdziora, adelton@fi.muni.cz

Tento článek byl inspirován debatou v linuxové a databázové konferenci o tom, na co vlastně potřebujeme transakce. Obvláště dotaz pana Váchy „pokud mi systém nepadá, chybové stavy testuji, existuje případ, že danou úlohu bez transakcí neudělám?“ nastolil zajímavý pohled na věc.

Podle manuálu Oraclu

> Transakce je logická jednotka zpracování dat, která se skládá z jednoho nebo více SQL příkazů provedených jedním uživatelem.

Podstatné je, že transakce končí buď commitnutím, tedy promítnutím změn do databáze, nebo rollbackem, vrácením databáze do původního stavu, resp. neprovedením změn naakumulovaných v průběhu transakce. Cílem tedy je, aby byly ostatními paralelně pracujícími uživateli vidět buď všechny změny, nebo žádná.

Triviálním postupem je v okamžiku zahájení transakce jedním uživatelem zablokovat přístup pro ostatní, tedy databázi či její část zamknout, to ale není v multiuživatelském prostředí únosné. Databázový server tedy musí být schopen dát v jednom okamžiku různým uživatelům různé stavy databáze, protože jeden už něco změnil a druhý nikoli. Tito uživatelé se nesmějí navzájem nijak negativně ovlivňovat, práce jednoho nesmí blokovat práci druhého. Pouze pokud se oba pokusí změnit ta samá data (a záměrně neříkáme, zda ten samý řádek tabulky či tu samou tabulku, jelikož granularita se může server od serveru lišit), je zpracování jedné transakce pozastaveno do doby, než druhá transakce skončí.

Násobné kopie různých verzí databáze jsou uloženy ve zvláštních oblastech, nazývaných rollback segmenty či snapshoty. Zde jsou typicky uloženy původní stavy databáze, což zajišťuje, že commit je rychlý, neboť je pouze poznamenáno, že transakce již skončila a pro paralelně pracující uživatele již není nutno rekonstruovat původní verzi databáze. Rollback pak naopak znamená delší čas na navrácení původních dat na původní místo.

V nejstriktnějším módu izolace jednotlivých transakcí musí uživatel vidět v průběhu transakce databázi v takovém stavu, v jakém byla při zahájení transakce. Tedy ani paralelně běžící již commitnuté změny nesmí být viditelné. ASNI SQL standard zná i uvolněnější módy, například kdy transakce vidí nové řádky, které přidala (a commitnula) jiná, paralelní transakce.

Pokud databáze podporuje transakce, je možno toho s výhodou použít i pro rekonstrukci konzistentního stavu databáze po pádu, například při výpadku proudu – všechny necommitnuté transakce jsou zrušeny a databáze vrácena do původního stavu, kdy ani jedna z transakcí není neukončená.

Většina databázových serverů dovoluje uživateli přepnout se do tzv. autocommit nebo-li unchained módu, kdy je každý SQL příkaz implicitně okamžitě commitnut na disk. Typicky ale i zde se pro zajištění paralelní práce více uživatelů používá výše popsaného mechanismu, neboť i zpracování jediného SQL příkazu může trvat dlouho a tedy i takto krátkou transakci je vhodné pro zvýšení celkového výkonu serveru ošetřit vestavěným mechanismem správy násobných verzí dat než uzamčením celé tabulky či části databáze.

Pokud máme databázový server, který transakce nepodporuje, například MySQL či MiniSQL, co to pro naši práci znamená? Pokud chceme zajistit konzistentní pohled všech uživatelů na data, tedy nechceme, aby paralelně pracující uživatel viděl, že peníze již odešly z jednoho účtu, ale ještě nedorazily na druhý, musíme po dobu provádění takovýchto atomických akcí složených z více SQL příkazů relevantní tabulky zamknout a tím přístupu ostatních uživatelů k vnitřně nekonzistentním datům zabránit. Dále musíme psát aplikace tak, že nejprve ošetříme/otestujeme všechny chybové stavy, které mohou nastat (vložení duplicitní hodnoty do sloupce s primárním klíčem, porušená referenční integrita) a pak teprve začneme provádět akce, o kterých v podstatě předem musíme vědět, že budou úspěšné.

Nutnost zamykání tak paradoxně může vést k tomu, že systém je celkově méně průchodný než pokud by sám implementoval transakční zpracování. Je proto dobré zvážit, jakou třídu aplikací chceme na serveru provozovat. Pokud je databázový server nasazen v relativně jednoduché WWW aplikaci, kde většina přístupů je čtení dat a velmi malé procento změna malého objemu dat, či kde není množství paralelně pracujících uživatelů měnících data velké, je MySQL více než dostačující. Mimo jiné proto, že pro uchování násobných verzí databáze je nutno počítat s vyšší režií serveru i s dodatečným diskovým prostorem.

Naopak pro kritické aplikace je nutností databáze, která například pomocí redo logů a garantovaného zápisu bloků fyzicky na disk je schopna ošetřit i hardwarový výpadek, příkladem může být Oracle. Někde mezi pak stojí databázové servery typu PostgreSQL, kde transakce jsou víta-

ným nástrojem zjednodušení práce aplikačního programátora, který nemusí řešit množství případných chybových stavů a může psát aplikaci přímočařeji, kde případná chyba je pouze následovaná rollbackem bez potřeby dodatečných testů. Transakce mohou (ale nemusejí!) zvýšit průchodnost serveru zvláště při mnoha paralelních změnách, protože ve většině případů eliminují nutnost zamykání větších částí databáze než jednotlivých řádků.

Závěr a možná odpověď na otázku z úvodu? Ano, obejdete se bez transakcí. Pokud máte databázový server zvládnutý a vyhovuje vám a vaší aplikaci, asi není důvod přecházet na jiný jen kvůli magickému slůvku transakce. Pokud máte pod kontrolou všechny aplikace, které na data sahají, může být řešení My rychlejší a flexibilnější než řešení Ora. Na druhou stranu, nikdy nezaškodí vyzkoušet oboje a rozhodnout se dle vlastního pohledu. ■

Zasmáli jsme se

Pavel Janík ml., 31. října 1999

Konečně jsem dnes dokončil dlouhý výběr z anekdot a vtipů, které se mi za uplynulý měsíc nashromáždily v mé přihrádce určené Linuxovým novinám, a tak jsem pro vás, čtenáře Linuxových novin, připravil výběr toho nejzajímavějšího a nejzábavnějšího, co se událo v linuxových konferencích a vůbec kolem Linuxu. Je toho tentokráte hodně, a tak se do toho hned pusťme.

Většina příspěvků, které zde zveřejňujeme, se týká pouze Linuxu, ale linuxoví uživatelé nejsou zase tak zaslepení, aby se zajímali pouze o Linux, a tak přinášíme i několik vtípků z konferencí o Perlu a Oracle (díky Adeltonovi):

V konferenci Perl-XML se diskuse trošku zvrhla a dopadla následovně:

> Like many coders, I work with both Perl and Java frequently. It's important that we co-operate so that we can fight the common enemy, Python (sorry, I mean Microsoft; no, it's C++; or is it the U.S. anti-encryption laws? I always get it mixed up).

Uživatelé Oraclu pod Linuxem se nenechali zmást zmateným uživatelem, který si myslel, že Oracle pro Linux poběží na platformě Wintel, a tak se mu dostalo zajímavé odpovědi:

> > > I believe that Oracle for Linux is only for Wintel machines. I know
> > > that I cannot get it to run on an Alpha.
> >
> > No, no, Oracle for Linux _doesn't_ work on
> > Wintel machines. That must
> > be a _Lintel_ machine if it runs Linux. ;-)

Problémy s dodržováním pravidel různých konferencí a skupin nemáme jenom my s naší skupinou, ale také ostatní. V konferenci comp.lang.perl.misc např. proběhlo:

> >For example, particularly shitty mail clients (Outlook not so
> >good! :-) choke on valid mboxes.
> >
> >This admittedly doesn't have much to do with Perl.
> I think the general policy is to consider Microsoft bashing always on topic. ;-)

Ale abychom nebyli pouze jednostranně zaměření, zveřejníme také jednu zajímavou básničku. Já jsem měl básničky rád již na gymnáziu – paní učitelka mne vždycky vyvolala a já mlčel. „Pět," ozvalo se z jejích úst. Když se to ozvalo i za týden, paní učitelka to se mnou vzdala a pochopila, že na básně musí mít člověk talent a že učit žáky matematické třídy rozumět poezii je velmi těžký úkol… Ale tuto se jistě naučím :-)

Na mém čipu značky Cirrus
usadil se ňákej virus.
Z VGA se na mě cení
a snížil mi rozlišení.

Jeho bráchů velký chór
obsadil mi procesor.
Ignorují Intel značku,
z Pentia mám kalkulačku.

Také disk se špatně točí,
koukám, div mi nevypadnou oči,
další velká virů síla
z kapacity mi ubírá.

Jiný velký virů sled
zpomalil mi Ethernet.
Mají z toho velkou bžundu:
čtyři bity za sekundu.

O paměti nemluvím,
tam jsou taky – já to vím.
Uspořádali tam párty,
žerou bity, hrajou karty.

Pár jich sedí na myši,
nevnímají, neslyší,
ať řvu jak chci – neštěstí,
kurzor prostě nejezdí.

Naděje však neumírá,
chytil jsem jednoho vira
a zjistil, že na zadečku
mikroskopickou má tečku.

Mikroskopem zkoumám hnedka,
od začátku, od prostředka,
a zírám, neschopen slov:
Copyright by Microsoft.

Deset sekund po té scéně
mažu Windows nenuceně
a na nyní volné místo
Linux dávám – to je jisto.

Od té doby dnes a denně
usmívám se potěšeně
a zůstávám stále svěží

a můj počítač – ten běží.

Když už jsme u těch virů – nedávno se na deníku Svět barevně (omlouvám se, ale zapomněl jsem tu správnu barvu, ale vím, že Narudo to nebylo) objevil článek, který komentoval objevení prvního viru pro Microsoft PowerPoint. Jeho název zněl: „Nabídka virů pro aplikace Microsoftu je nyní kompletní“ :-)

Dnes jsme tu ještě neměli signatury. Tak jich hned několik přidám k dobru.

> “[Open Source] programming is like sex, one mistake and you have to support it for the rest of your life.” M. Sinz, CBM Inc.

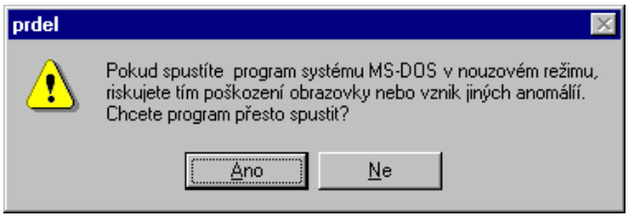
(Něco na tom bude :-)).

> This is Linux Country. On a quiet night, you can hear Windows reboot!

A nakonec jedno staré přísloví, samozřejmě poněkud upravené:

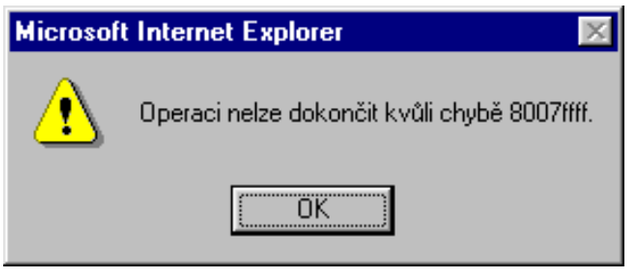
> Neuč orla létat a MS Windows padat.

Tak a to je ode mne vše. Ještě přídáme několik krásných hlášek jednoho neznámého operačního systému, které připravil a nachytal Luboš Němec, a budeme se těšit na další rubriku, kterou připravujeme ve spolupráci s vámi, našimi čtenáři. Proto neváhejte a vše, co vás pobavilo či rozesmálo, nám zašlete.



Takhle to dopadá, když se člověk snaží spustit MS-DOS program „prdel“, což je i na Windows trošku moc.

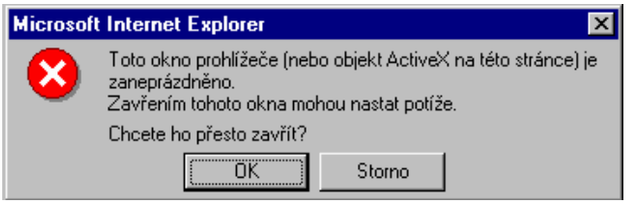
Corel Draw, to je kapitola sama pro sebe, jenom by mne zajímalo, kolik že to vlastně příkazů nakonec uspěje.



Že by slabé nervy?

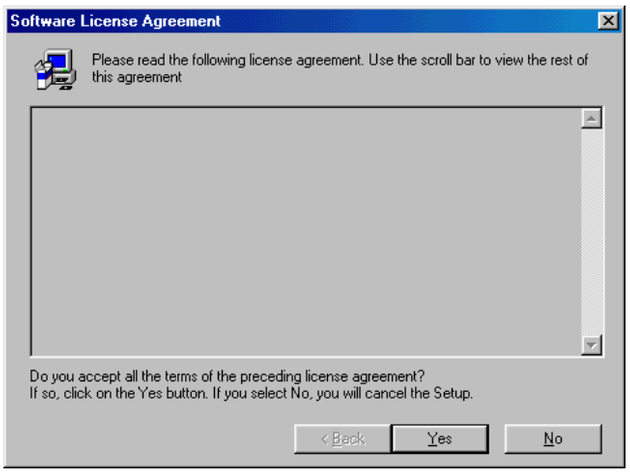
Opět skvěle vystižená specifikace chyby.

Opravdu důrazné varování!

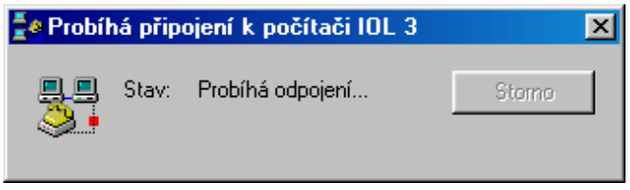


Jen jedna rada – nespouštějte Explorer 3 a nenastanou potíže.

Ano, chápu, my vsichni byt spravna čecha a dukladne rozumet.

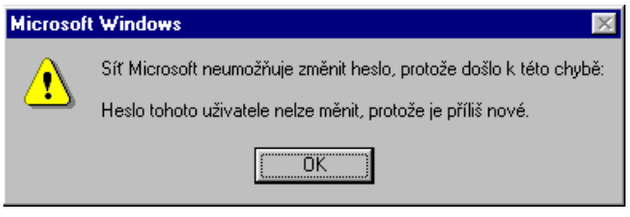


Takhle to dopadá, když chcete nainstalovat jednu z verzí QuickTime pro Windows. Jen upozorňuji, že pokud NEBUDETE souhlasit s licenčním ujednáním, nelze program pochopitelně nainstalovat. Hmm, a s čím že to vlastně mám souhlasit?



Tak probíhá připojení nebo odpojení? Dělat z lidí blbce, to je u Windows normální.

Jo, ta chyba byla opravdu velmi osudná a pak to celý lehlo.



Ta chyba je vážně tak hrozná, že mám pochybnosti o zdravém rozumu programátorů, klasický případ toho, jak hloupost vládne světu… ■

Linuxové noviny a jejich šíření

Linuxové noviny vydává České sdružení uživatelů operačního systému Linux (1) pro své příznivce a sympatizanty. Vlastníkem autorských práv k tomuto textu jako celku je Pavel Janík ml. (Pavel.Janik@linux.cz). Autorská práva k jednotlivým článkům zůstávají jejich autorům.

Tento text může být šířen a tištěn bez omezení. Pokud použijete část některého článku zde uveřejněného v jiných dílech, musíte uvést jméno autora a číslo, ve kterém byl článek uveřejněn.

Linuxové noviny jsou otevřeny každému, kdo by chtěl našim čtenářům sdělit něco zajímavého. Příspěvky (ve formátu čistého textu v kódování ISO 8859-2) posílejte na adresu (2). Autor nemá nárok na finanční odměnu a souhlasí s podmínkami uvedenými v tomto odstavci. Vydavatelé si vyhrazují právo rozhodnout, zda Váš příspěvek uveřejní, či nikoli.

Registrované známky použité v tomto textu jsou majetkem jejich vlastníků.

Chtěl bych poděkovat Fakultě informatiky Masarykovy university v Brně, INET, a.s., Juraji Bednárovi, Milanu Šormovi za poskytnutí diskového prostoru pro Linuxové noviny.

Linuxové noviny můžete najít na akademické síti TEN-34 CZ (3), na síti Global One na adrese (4), na serveru Gymnázia Vídeňská v Brně (5) a na serveru časopisu Netáčik (6), který je připojen do slovenského SIXu.

Linuxové noviny jsou k dispozici také ve formátu HTML na adrese (7). ■

1 České sdružení uživatelů operačního systému Linux
http://www.linux.cz/czlug
2 Adresa redakce
mailto:noviny@linux.cz
3 Linuxové noviny na síti TEN 34-CZ
ftp://ftp.fi.muni.cz/pub/linux/local/noviny
4 Linuxové noviny na síti IBM Global One
ftp://ftp.inet.cz/pub/People/Pavel.Janik/noviny
5 Linuxové noviny na komerční síti CESNET
http://www.gvid.cz/linux/noviny/
6 Slovenské zrcadlo Linuxových novin
ftp://netacik.sk/pub/linux/cz-noviny
7 Linuxové noviny ve formátu HTML
http://www.linux.cz/noviny

Šéfredaktor: Pavel Janík ml.
mailto:Pavel.Janik@linux.cz
sazba: Ondřej Koala Vácha
mailto:koala@informatics.muni.cz
jazykové korekce: Bohumil Chalupa
mailto:bochal@met.mff.cuni.cz
překlady: Hanuš Adler
mailto:had@articon.cz
převod do HTML: Pavel Juran
mailto:juran@proca.cz