08–09/98

Linuxové noviny

Úvodem

Pavel Janík ml., 1. září 1998

Právě je jedna hodina po půlnoci, vypínám svoji Sněhurku a moje šestiletá dcerka se ráno chystá poprvé do školy a já jsem nervóznější než ona. Sešity už má nachystané, velký kornout s denní dávkou sladkostí také, chybí jenom svačina, kterou maminka nestihla večer nachystat…

Tak takhle nějak mohl vypadat dnešní den i u někoho z vás. Pokud ano, tak zapomeňte na Linuxové noviny, zapomeňte na Linux a věnujte se malé dcerušce. Až dceřušku odvedete do školy, pohodlně se usaďte, otevřte (láhev beaujolais, …) nové číslo Linuxových novin a začněte číst.

Tohle číslo je výjimečné hned kvůli několika věcem. První z nich jsou dva zajímavé články, které napsaly dvě sličné dívky. Druhou výjimečnou věcí je to, že toto číslo je vlastně dvojčíslem. Všichni jistě chápete, co nás k tomu vedlo. Ano, byly prázdniny a doba dovolených a i autoři Linuxových novin potřebují někdy zregenerovat své síly, aby mohli i nadále přinášet zajímavé informace ze světa operačního systému Linux.

A co se vlastně dočtete? Michal Choura nám ukáže, že SGML opravdu není tak strašný, ve svém článku Jak na Linuxdoc-sgml. Vláďa Michl nás provede vlákny a jejich použitím v článku Linux a vlákna.

V článku Báječný svět jádra v. 2.2 najdeme první díl dlouhého článku o tom, co se změnilo v jádře Linuxu od verze 2.0. Autorem překladu je Nathan L. Cutler. Martin Mareš nám ve svém článku Linux 2.1 a sítě představí novinky v podpoře síti ve vývojových verzích jádra.

Michal Fadljevič se nám ve svém seriálu pokusí přiblížit, jak začít s editorem Emacs Začínáme s Emacsem: I — Spuštění a ukončení.

V článku Jaký byl Linux InstallFest? se vám pomocí několika fotografií pokusím přiblížit, jaký vlastně byl první Linux InstallFest pořádaný v naší republice.

Novinky na serveru *sunsite.unc.edu* najdete jako v každém čísle Linuxových novin v článku Co nového na sunsite.unc.edu? a průřez příspěvky ve skupině COLA (*comp.os.linux.announce*) je v článku Měsíc v comp.os.linux.announce.

Přeji příjemné čtení a těším se na vaše připomínky na adrese redakce (1). ■

1 Adresa redakce
mailto:noviny@linux.cz

Magické dny pro Linux?

Pavel Janík ml., 1. září 1998

Když jsem připravoval nové číslo Linuxových novin, postupně jsem si tiskl dokumenty, které jsou zajímavé a o kterých bych mohl něco napsat. Můj původní záměr byl do úvodníku zahrnout úvahu na téma: „Linux ve světě velkých aplikací“, protože v průběhu července došlo ke třem velmi zajímavým událostem. Ale začněme od začátku.

21. července vydala společnost Netscape Communications Corp. (1) tiskovou zprávu, ve které oznamuje své plány portovat své programové vybavení pro servery na operační systém Linux a umožnit tak poskytovatelům připojení provozovat servery Netscape i na Linuxu. Netscape Messaging Server a Netscape Directory Server by měly být k dispozici v první čtvrtině roku 1999.

Ale 21. července to ještě nebylo vše. Podobné prohlášení vydala i společnost Oracle (2). Jejich databázový server Oracle8 bude k dispozici ještě na konci tohoto roku. Budeme si dokonce moci vyzkoušet verzi, jejíž licence bude omezena na 90 dní. Myslím, že společnost Oracle udělala velice důležitý krok pro to, aby se stala nejúspěšnějším producentem databázového vybavení pro operační systém Linux.

Den poté — tedy 22. července — si další společnost, tentokráte Informix Corporation (3) uvědomila, že by mohla zůstat pozadu za Oraclem a tak nejen ohlásila portování, ale dokonce již uvolnila Informix-SE pro Linux.

Jak vidíte, náhody se dějí (tím mám samozřejmě na mysli shodné datum, nikoli to, že tak úspěšné společnosti se zabývají Linuxem) a doufejme, že se objeví i další. Na Internetu kolují zprávy např. o tom, že by společnost IBM mohla portovat nějaké své aplikace a dokonce i o tom, že už něco mají hotovo, ale nechme se překvapit.

Magické to dny, jen více jich … ■

1 Netscape Corp.
http://www.netscape.com
2 Oracle Corp.
http://www.oracle.com
3 Informix Corp.
http://www.informix.com

Měsíc v comp.os.linux.announce

Pavel Janík ml., 1. září 1998

V konferenci *comp.os.linux.announce* je co číst i o prázdninách, a tak vám přinášíme i nyní výběr toho nejzajímavějšího.

Společnost Linux Central (1) vydala novou knihu pro programátory pod názvem Linux Programmer's Reference. Autorem je Richard Petersen. Bližší informace o knize samotné naleznete na adrese (2).

Potřebujete konvertovat textové dokumenty do formátu PDF a nechcete používat pdfTEX? Potom právě pro vás je určen program `txt2pdf`, který najdete na adrese (3).

Světlo světa spatřil další office-like balík pro Linux. Jeho jméno je Siag Office a je samozřejmě šířen pod licencí GPL. Více informací naleznete na adrese (4).

Markku Rossi (mtr@ngs.fi) oznámil nový interpreter Javascriptu — NGS JavaScript Interpreter (5).

4Front Technologies (front@best.com) portovala svůj

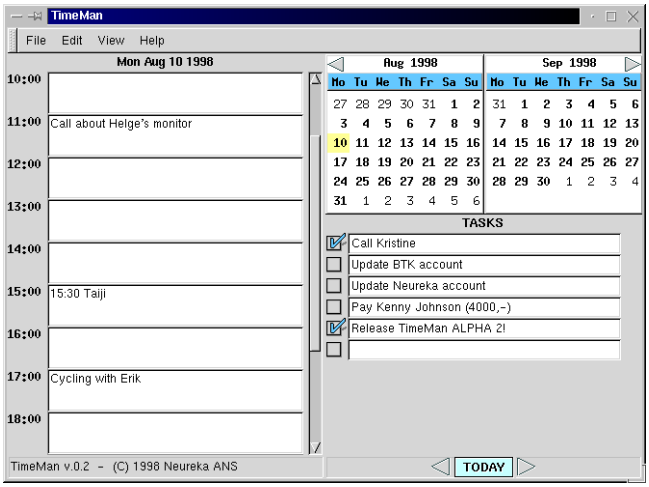
Open Sound System v3.9 i na pracovní stanice Alpha. Více informací získáte na adrese (6).

Stan Shebs (shebs@cygnus.com) oznámil novou verzi hry Xconq (7).

Tigran Aivazian (tigran@einstein.london.sco.com) vytvořil zajímavý modul do jádra Linuxu (Time Travel), který umožňuje — jak již název napovídá — cestovat v čase. Je vhodný např. pro testování odolnosti vašich programů proti roku 2000 apod. Najdete jej na adrese (8).

Společnost Transvirtual Technologies, Inc. uvolnila svoji implementaci Java VM pod názvem Kaffe OpenVM včetně kompletních zdrojových textů. Další informace o tomto záslužném činu naleznete na adrese (9).

H. J. Lu zveřejnil poslední verzi knihovny `libc` verze 5 — 5.4.46. Knihovnu najdete na adrese (10).

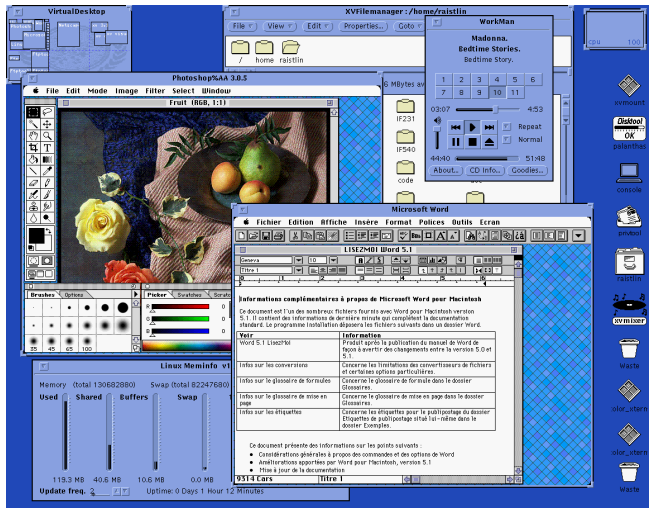


Arne O. Morken (arnemo@ii.uib.no) vytvořil program TimeMan, který jistě mnohým z našich čtenářů nahradí jejich ADK. TimeMan naleznete na adrese (11).

Linux Quake Howto naleznete na adrese (12).

Společnost ARDI uveřejnila novou verzi svého emulátoru prostředí Macintosh — Executor (13). Executor umožňuje spouštět binárky pro Macintosh — mezi podporované programy patří např.: Microsoft Word, Microsoft Excel, Adobe Illustrator. ■

1 Linux Central
http://linuxcentral.com
2 Linux Programmer's Reference
http://linuxcentral.com/products/book/progref/
3 txt2pdf
http://www.geocities.com/CapeCanaveral/Hangar/4794/
4 Siag Office
http://www.edu.stockholm.se/siag/
5 NGS JavaScript Interpreter
http://www.ngs.fi/js/
6 Open Sound System
http://www.opensound.com
7 Xconq
ftp://ftp.cygnus.com/pub/xconq/
8 Time Travel
http://www.aivazian.demon.co.uk/tt/tt.html
9 Kaffe OpenVM
http://www.transvirtual.com/
10 Knihovna libc 5.4.46
ftp://sunsite.unc.edu/pub/Linux/GCC
11 TimeMan
http://www.bgif.no/neureka/TimeMan/TimeMan.html
12 Linux Quake Howto
http://quake.medina.net/howto
13 Executor
http://www.ardi.com



Co nového na sunsite.unc.edu?

Pavel Janík ml., 1. září 1998

X11

X11/desktop/xnc-4.0.5.src.tar.gz — XNC desktop

X11/devel/builders/mgui-2.0.1.tar.gz — rozhraní s okny pro DOS, Windows, and X

X11/xutils/xgreekeychanger-1.3.0.tar.gz — umí měnit mapu klávesnice pod X Window System

apps

apps/database/postgresSQL/pe-0.96.tgz — editace formulářů PostgresSQL v Emacsu

apps/doctools/man/man-1.5e.tar.gz — prohlížeč manuálových stránek

apps/editors/X/GXedit-1.08.tar — jednoduchý editor založený na knihovně Gtk

apps/graphics/wisio-0.5.tgz — grafický okenní server pro VGA

apps/sound/cdrom/X/xfreecd-0.7.7.tar.gz — přehrávač zvukových CD disků s rozhraním Gtk

apps/wp/maxwell-0.5.3-src.tar.gz — WYSIWYG textový procesor

commercial

commercial/visualmail-3.0s.tar.gz — visuální mailer

distributions

distributions/small-linux-v0.20.tar.gz — mini distribuce Linuxu na dvou disketách

games

games/strategy/phalanx-16.tar.gz — český šachový program pro Linux

kernel

kernel/getpatch-2.8.tar.gz — skript ulehčující stahování patch souborů

logos

logos/penguins/linux_logo-2.05.tar.gz — pokud chcete mít na své obrazovce barevného tučňáka, neváhejte

system

system/network/management/fwconfig.tar.gz — konfigurace Linuxu jako firewallu pomocí `ipfwadm` (GUI)

■

Linux Journal — červenec

Petr Bárta, 4. srpna 1998

V červencovém čísle Linux Journalu se dozvíte o tom, jak je možné Linux využít v praxi — konkrétně ve vědecké a inženýrské práci.

Autory prvního z článků věnovaných tomuto tématu jsou P. Klosowski, N. Maliszevskyj a B. Dickerson, zaměstnanci amerického Institutu pro standardy a technologie. Tématem je použití Linuxu při sběru a zpracování dat ve vědeckém prostředí (National Institute of Standards and Technology).

F. Rodemaker a R. Burn představují *ROOT* — systém pro zpracování, analýzu a prezentaci velkého množství dat, vyvíjený v CERNu.

Částečně z CERNu je i další článek, tentokrát věnovaný použití Linuxu při proměřování vlastností krystalů používaných v urychlovačích. S podrobnostmi seznamují E. Leonardi a G. Organtini.

O tom, že nejenom na souši, ale i při námořních výzkumech se uplatňuje Linux, nás informují C. Donlon a J. Crawshaw. Konkrétním příkladem je britská výzkumná loď *James Clark Ross*.

R. Sevenich a R. Prize přináší článek o využití „fuzzy“ logiky. Jejich program na odhad lavinového nebezpečí využívá právě tento algoritmus, byl napsán v Javě a na Debian/GNU Linuxu.

GPS (*Global Positioning System*) byl vyvinut hlavně pro vojenské účely. Že se ale dá použít i jinak, nás přesvědčí R. Parry ve svém článku o APRS (*Automatic Position Reporting System*).

V rubrice „Kernel Korner“ vysvětluje A. Rubini, jak napsat a použít svůj vlastní „misc“ driver do jádra.

Znáte *Icon*? Pokud ne, přečtěte si článek C. Jefferyho a S. Mohameda, kteří přinášejí stručný úvod k tomuto jazyku.

Bezpečnosti se věnuje B. Giles, v tomto článku vysvětluje, jak se postarat o zašifrování ne jednotlivých souborů, ale celého filesystému.

Další z programovacích jazyků, který je v tomto čísle Linux Journalu představen, je Graphical Desktop Korn Shell (DtKsh), součást Common Desktop Enviroment (CDE). Představuje ho G. Kraft.

Seznámit se s balíkem *Samba* můžete (tedy ti, kteří ho ještě neznají) prostřednictvím článku J. Blaira.

V rubrice „Take command“ uvádí D. A. Provins příklad použití dvou méně známých příkazů — `lex` a `yacc`.

Pohled na třetí programovací jazyk v tomto čísle — *Yorick* — přináší C. O'Brien v článku převzatém z Linux Gazette.

R. M. Lerner se v další pravidelné rubrice „At the Forge“ věnuje kombinaci Apache a PERLu — modulu `mod_perl` a jeho použití.

Samozřejmě i v tomto vydání Linux Journalu najdete ještě několik dalších článků, spoustu reklam, nějaké ty recenze a informace o nových produktech.

Na závěr bych se rád omluvil za překlep v této rubrice v minulém čísle Linuxových novin — týkala se červnového čísla Linux Journalu, a nikoliv červencového, jak bylo nesprávně uvedeno. ■

Linux Journal — srpen

Petr Bárta, 9. září 1998

Srpnové číslo Linux Journalu podle mého názoru nemá jedno hlavní téma. Ostatně máte možnost posoudit sami.

První, velmi rozsáhlý článek, resp. rozhovor, je věnován firmě Netscape, její strategii přihlášení se k filosofii Open Source, a problematice Open Source obecně. Doc Searls, Mark Andreessen a Tom Paquin diskutují o směrech rozvoje software průmyslu, důvodech, které vedly k rozvoji Open Source filosofie, o důvodech a důsledcích uvolnění zdrojového kódu Netscape prohlížeče a uplatňování Darwinových metod přirozeného výběru v produkci programů.

Článkem Normana M. Jacobowitze začíná nový seriál, ve kterém autor popisuje své osobní zkušenosti s přechodem k Linuxu. V tomto dílu se věnuje rozdílům v instalaci,

rychlosti učení, ceně, stabilitě a nákladech na provoz mezi Linuxem (použitá distribuce Red Hat 5.0) a Windows NT 4.0.

Pokud nevíte, kterou distribuci si vybrat pro svůj počítač, zkuste si přečíst porovnání P. Hughese. Autor srovnává jednotlivé distribuce, se kterými se setkal, kromě toho je k článku připojena zajímavá srovnávací tabulka.

Hlavně začátečníkům, toužícím po větším množství dokumentace, je určen článek M. Stutze, shrnující jednotlivé metody, jak se v případě problémů dobrat nápovědy a řešení.

Nevíte, jak nastavit svůj server jako záložní poštovní počítač pro případ výpadku hlavního poštovního počítače ve vaší firmě? Pak si nenechte ujít článek J. Blaira právě na toto téma.

M. Telgarsky popisuje standardní rozdělení adresářů v Linuxu v článku věnovaném opět hlavně začátečníkům.

Informace o projektu *MUSCLE*, jehož autoři se snaží o integraci tzv. „smart cards“ do Linuxu, se dozvíte z článku D. Corcorana.

Zajímavý článek o tom, co je *XSuSE*, proč vznikl a jak souvisí s XFree86, napsal viceprezident projektu XFree86, D. H. Hohndel.

D. Haraburda seznamuje s novou distribucí — *Linux Stampede*.

Zajímá vás problematika síťových clusterů a rychlé výměny dat mezi jednotlivými členy těchto clusterů? Potom je článek B. Ellistona o zapouzdření IP do protokolu SCSI určen právě pro vás.

Vytváření grafů pro WWW stránky v reálném čase na vyžádání se věnuje M. Pruett.

Pro zobrazení výstupu z databází na WWW serverech můžete použít mimo jiné i modul pro server Apache — *mod_perl*. Ve volném pokračování článku z minulého čísla Linux Journalu se R. Lerner věnuje modulu DBI, DBD a Apache::DBI.

A opět, jako vždy, je v Linux Journalu spousta inzerátů, recenzí, reportáží a informací o nových produktech. Zaujala mne recenze na knihu *The No B.S. Guide to Linux* B. Rankina nebo článek o využití Linuxu a OCR při běžné práci americké pošty (té šnečí `:-)`). ■

Linux 2.1 a sítě

Martin Mareš, 10. září 1998

Úvod

Již přibližně dva roky probíhá práce na vývojové řadě linuxových kernelů 2.1 a v současné době směřuje k vyvrcholení v podobě stabilního release 2.2. Za tuto dlouhou dobu se změnily spousty věcí v prakticky všech koutech jádra, nevyjímaje ani síťové subsystémy, ba naopak značné procento změn bylo právě v nich.

Tento článek se snaží formou alespoň trochu přehlednou čtenáři osvětliti, co nového přináší Linux 2.1 (respektive co přinese Linux 2.2) ohledně sítí ve srovnání s předchozí stabilní verzí 2.0. Původně byl tento text přednesen v květnu 1998 jako referát v rámci Linuxového semináře MFF UK (1). Zde se autor snaží o poněkud systematičtější a úplnější podání téhož tématu.

Při rozsahu síťové podpory v Linuxu a zejména pak při rozsahu diskutovaných novinek jsou pouze dvě možnosti, jak tento článek může dopadnout: buďto bude neúplný (prostě se o méně důležitých či méně zajímavých věcech nebude zmiňovat) a nebo bude sáhodlouhý, nudný a většina čtenářů jej před polovinou odloží jako vyčerpávající, a to nejen téma. Pokusil jsem se vybrat si variantu první jakožto menší z obou zel.

TCP/IP

V oblasti TCP/IP se toho změnilo snad nejvíce. Prakticky celý kód obhospodařující tuto sadu protokolů se dočkal přepsání od základu (někdy i několikanásobného), obvykle zásluhou největšího ze síťových mágů Alexeje Kuzněcova.

Jelikož klasické UNIXové (z BSD UNIXu pocházející) příkazy již dlouhou dobu nepostačují (nejen implementačně, ale zejména svou „dinosauří“ koncepcí neberoucí příliš na vědomí dávno již běžné věci jako je např. *classless routing* ke konfiguraci všech parametrů linuxového TCP/IP, vznikla nová utilita krátkého a výstižného jména `ip` (2) syntaxí v mnoha ohledech podobná příkazům routerů firmy Cisco (3) a pro používání velice šikovná. Leč bohužel zatím bez jediného řádku dokumentace, nepočítaje v to zabudovaný help popisující syntaxi příkazů.

Třídy adres (v dnešním světě mající význam čistě historický) již nejsou v kernelu podporovány, při zakládání routy či konfiguraci interfacu je již **nutno** zadat vedle adresy také netmask či délku prefixu.

Četné parametry protokolů je nyní možno nastavovat přes `sysctl`, případně pomocí interfacu v `/proc/sys/net` přímo ze shellu. Viz dokumentace od utility `ip :-)`

Socket Hashing

Před delší dobou se k uším Davida Millera doneslo, že FreeBSD má několikanásobně rychlejší přiřazování příchozích packetů ke spojením než měl tehdejší Linux. Po cca týdnu práce vznikl nový kód pro socket hashing, který tyto věci řeší několikanásobně rychleji než FreeBSD.

IP Aliasing

Zcela se změnila podpora aliasů zařízení (původní `eth0:1` apod.). Dnes již původní aliasy neexistují, leč každá síťová karta může mít mimo primární IP adresy ještě libovolně mnoho adres sekundárních. Původní interface s dvojtečkami ve jménech zařízení byl zachován pro kompatibilitu, nicméně v detailech se mírně liší.

IP Routing

Linux již nemá jedinou routovací tabulku, ba naopak tabulek může existovat více a za pomoci konfigurovatelných pravidel je možno vybírat pro každý příchozí packet, podle které tabulky bude routován, a to na základě zdrojové a cílové adresy a dalších parametrů, jako je například TOS (*Type Of Service*) a interface, ze kterého packet přišel. Tento mechanismus umožňuje velice pěkným způsobem implementovat jak *policy based routing*, tak *routing by TOS* i spoustu jiných triků.

Rovněž při přiřazení IP adresy interfacu automaticky vznikají device routy a při odkonfigurování samy mizí. Toto se připravovalo již pro kernel 2.0, nicméně kvůli problémům s kompatibilitou to bylo odloženo o release později.

K dispozici je *Equal Cost Multipath*, to jest distribuce packetů mezi několik ekvivalentních položek v routovací tabulce.

TCP

TCP bylo rovněž prakticky od základu přepsáno, z většiny zásluhou Davida Millera. Nyní má o dost lepší logiku timeoutů a retransmitů a zejména podporuje několik nově standardizovaných rozšíření:

- SACK (*Selective Acknowledge*) — výhodný pro linky mající vysokou chybovost.
- *Window Scaling* — užitečný na spojích, které jsou rychlé, ale přitom mají značné zpoždění (typicky například satelitní kanály).
- *Timestamps* — napomáhají lépe poznat charakter spoje a díky tomu též zvolit lepší strategii timeoutů.

Firewalling

Stejně jako verze 2.0, i 2.1 přináší zcela přepracovaný systém firewallingu. Do kernelu byla konečně zařazena podpora *IP Firewalling Chains* (4) umožňující řetězení filtrů (což je jednak rychlejší, jednak snazší na údržbu). Tato změna opět není zpětně kompatibilní, firewall je nyní nutno konfigurovat novým programem `ipfwchains`. Podrobnosti viz článek v čísle 3/1998 Linuxových novin.

Masquerading

Přibyla podpora pro překlad adres v dalších protokolech, rovněž pak možnost ošetřovat překlad protokolů prostřednictvím démona běžícího mimo kernel (viz balík `ipautofw` (5)).

Masquerading se nyní konfiguruje pomocí programu `ip-masqadm` (6).

K dispozici je též *Port Forwarding* (7) umožňující přesměrovávat vybraná TCP spojení na zadaný stroj a zadaný port (užitečné například tehdy, chcete-li mít WWW server za firewallem na počítači s interní adresou a forwardovat všechna spojení s portem 80 firewallu na tento počítač). Konfiguruje se programem `ipportfw` (8).

Network Address Translation (NAT)

V souvislosti s novým routovacím kódem je podporováno překládání IP adres, prozatím pouze případy s navzájem jednoznačným přiřazením. (Tedy můžete si kupříkladu nechat překládat část adres interní sítě na veřejně dostupné adresy bez rekonfigurace interních počítačů.)

Plánuje se sjednocení tohoto mechanismu s masqueradingem, což přinese možnost jak statického, tak dynamického překladu libovolného počtu adres na libovolný jiný počet.

Tunneling

Linux se též naučil lépe tunelovat (packety, nikoliv banky — zdá se, že studiem kvantové ekonomie se linuxoví hackeři dosud nezabývají). K dispozici je jak klasický protokol *IP in IP* (nyní ovšem výrazně lépe implementovaný), tak GRE (*Generic Routing Encapsulation*, umožňuje tunelování různých protokolů, *multicasting*, *soft state management* apod.) a SIT (*Simple Internet Transition*, přenos IPv6 přes IPv4).

Fast Switching

Tato značně experimentální feature umožňuje přímý přenos packetů mezi buffery jednotlivých síťových karet bez „přestupu“ v hlavní paměti, což znatelně urychluje routing. Autor (již zmiňovaný Alexej Kuzněcov) tvrdí, že pak jeho vesměs normální PC má jen o 30% menší propustnost při forwardování než routery Cisco řady 7200. Bohužel zatím je tato vymoženost dostupná pouze se síťovými kartami vybavenými chipsetem Digital 21140 (DECChip Tulip) či 8390 (nutná speciální podpora v driveru).

IP Autoconfiguration

Automatická konfigurace IP prostřednictvím protokolů BOOTP či RARP již není spojena s NFS-rootem a lze ji používat i samostatně.

Netlink

Netlink je nový mechanismus pro komunikaci mezi uživatelskými programy (nejčastěji routovacími démony) a síťovou vrstvou. Umožňuje mimo jiné:

- Přenos změn v routovací tabulce, takže již není nutné periodicky scanovat celou (možno i několikamegabytovou) tabulku.
- Dozvídat se o událostech typu interface up, interface down apod.
- Přenos packetů, které neprošly firewallem (např. pro detailní analýzu útoků).

IP version 6

Linux 2.1 podporuje nový protokol IPv6 (9), který by měl během pár let zvolna začít nahrazovat původní IPv4 a zbavit tak svět problémů spojených s růstem Internetu.

IPv6 v kernelu funguje bez větších problémů (i když tento protokol samozřejmě dosud není tak vyladěný jako IPv4 a je v neustálém vývoji), ovšem na slušnou podporu v aplikacích si budeme muset ještě nějakou dobu počkat.

Viz též Linux IPv6 FAQ/HOWTO. (10)

Berkeley Packet Filter (BPF)

Přeci jen jsme se v něčem inspirovali systémem BSD: *Generic Packet Filtering*. Tato feature umožňuje uživatelským programům posílat kernelu packetové filtry v jednoduchém byte-kódu a tak si vybírat, které packety je zajímají a které ne. Nezbývá než doufat, že se `tcpdump` a podobné programy tuto vymoženost brzy naučí používat, a nebudou tak ztrácet drahocenný čas procesoru přenášením packetů, které samy po triviální analýze zahodí.

Drivery

Přibyla spousta nových driverů síťových karet a ještě větší

množství jich bylo aktualizováno. Viz stránka o driverech (11) od Donalda Beckera.

Též se začínají objevovat drivery na některé synchronní sériové karty, zejména pak zásluhou Alana Coxe.

Soundmodem

Příznivci packet radia nyní mohou používat softwarové packet modemy využívající zvukovou kartu jako A/D a D/A převodník. Za tuto možnost Linux vděčí Thomasi Sailerovi. Zdrojové texty soundmodemů jsou velice zajímavým čtením…

Na druhou stranu Linux dosud nepodporuje (a po delší diskusi na mailing-listu kernelových vývojářů to vypadá, že ani podporovat nebude) tak řečené WinModemy, což jsou „vylevněné“ částečně softwarové modemy pro běžné telefonní linky. Hlavním důvodem je notorická neochota firem tyto modemy vyrábějící k jakékoliv spolupráci.

Syscall sendfile

Přibyla nová systémová služba umožňující kopírování dat mezi souborem a libovolným file-handlem. Nejedná se sice o čistě síťovou záležitost, nicméně právě u síťových aplikací to může být velice užitečné (např. FTP server posílající data přímo ze souboru do socketu bez kopírování přes adresní prostor FTP daemona).

Síťové filesystémy

NFS

Implementace NFS (*Network File System* firmy SUN) doznala značných změn. Nyní má podstatně lepší cacheování a též by měl být schopen fungovat přes TCP.

Na druhou stranu, NFS je sice podporováno v UNIXovém světě prakticky každým systémem, leč je to neobyčejně mizerně navržený protokol, který již dlouhá léta čeká na někoho, kdo bude mít dost odvahy na to, aby jej nahradil něčím lepším. Máte-li o práci na podobném projektu zájem, neváhejte a ozvěte se autorovi tohoto článku (12).

Kernel NFS Server (knfsd)

Olaf Kirch vytvořil NFS server běžící v kernelu. Na první pohled by se mohlo zdát, že to je krásný příklad něčeho, co se má řešit jako user-level daemon, ale bohužel, Olafovo řešení je podstatně rychlejší (ušetří se kopírování všech dat mezi kernelem a adresním prostorem příslušného démona). Na druhou stranu, možná by stálo za zamyšlení, jestli by se podobné rychlosti nedalo dosáhnout šikovným použitím (a případně rozšířením) mechanismu `sendfile`.

CODA

Linux nyní má i implementaci distribuovaného filesystému CODA. Bližší informace viz stránky projektu CODA (13).

Automounter (autofs)

Konečně důstojná náhrada za starého auto-mount daemona (`amd`), to jest něco, co umožňuje *spolehlivě* automaticky mountovat disky při prvním pokusu o přístup do daného adresáře.

Svět Novellu

Stran podpory sítí světa novellského máme pouze dvě novinky: jednou je podpora protokolu SPX (*Sequence Packet Exchange*) a druhou pak všemožná vylepšení NCPFS.

Poměrně podrobné povídání o těchto věcech je v květnových Linuxových novinách.

Packet Socket

Původní metoda přímého přístupu k packetům (opět podivné dědictví z archaických UNIXů) byla nahrazena metodou rozumnou a konsistentní, konkrétně zavedením nového protokolu `PF_PACKET`, který umožňuje přímo vysílat i přijímat jak kompletní link-level packety, tak packety zbavené linkových hlaviček.

Důsledky: `tcpdump` již nemusí podporovat všechny existující linkové vrstvy — když link layer nepozná, prostě si nechá packety posílat „rozbalené“ a obsahu porozumí. Konečně je možné čistě implementovat servery, jakož i klienty pro BOOTP a DHCP a neuchylovat se kvůli tomu k podivným trikům typu přiřazování nulové IP adresy interfacu, což stejně s kernelem 2.1 nefunguje.

Packet Scheduling

Významným přínosem je zavedení nastavitelných a vyměnitelných packet queue managerů pro jednotlivá síťová zařízení. Pro každý interface si nyní můžete vybrat, jak přesně bude spravována jeho odchozí fronta a jak se budou řešit případy, kdy místo na packety bude docházet. Tento mechanismus řeší hned několik problémů najednou:

- QoS (*Quality of Service*) — spojení o zaručené propustnosti, prioritní toky, spolupráce s protokolem RSVP a jiné vymoženosti.
- *Load Balancing* — distribuce zátěže mezi několik interfaců vedoucích stejným směrem, náhrada za původní *equalizer device* a rozumná alternativa k *Equal-Cost Multipath*. Řeší se velice jednoduše: interfacům, mezi něž chceme rozdělovat tok dat, nastavíme speciální queue mód a vytvoříme virtuální interface, který bude fungovat jako zdroj dat pro tuto frontu.
- TOS — priorita packetů je automaticky zvolena na základě jejich *Type Of Service*.

K tomuto všemu je ještě nutné přidat algoritmus, který packetům přiděluje jejich priority. Můžete si vybrat z následujících možností:

- klasifikace dle routovacích pravidel — každé z pravidel pro výběr routovací tabulky může rovněž volit prioritu.
- klasifikace dle firewallu — firewallové tabulky mohou rovněž přidělovat prioritu packetů.
- RSVP — klasifikace dle údajů zasílaných RSVP daemonem.
- U32 — přímé matchování dat v hlavičkách packetů. Universální, ale pomalejší než ostatní možnosti.

Vše se konfiguruje novou utilitou `tc`, dokumentace jako obvykle zatím žádná.

Acorn Econet

Podporován jest i Econet, podivná to proprietární síť navržená a používaná firmou Acorn, a to jak po původních kartách od Acornu, tak po jiných sítích (*dárkové balení* v UDP).

WAN Router

Linux 2.1 rovněž obsahuje kostru kódu pro WAN routing — na ni se mohou „přivěšovat“ modulární drivery pro Frame Relay a jiné WAN protokoly.

Viz stránka o X.25 a Frame-Relay síťování (14).

X25 a LAPB

Ti nešťastníci, kteří potřebují používat síťě založené na protokolu X25 (opravdu je lituji, něco tak mizerně navrženého jsem snad v životě neviděl … možná s výjimkou NFS `:-)`), naleznou jistou podporu tohoto archaismu, čítaje v to i LAPB (nižší vrstva určená pro spolehlivý přenos datagramů „na dostřel“).

Network Block Device

Pavel Machek stvořil driver, který po TCP spojení emuluje blokové zařízení (zatím bez `ioctl` a podobných fines, nicméně vcelku použitelně). Viz domácí stránka projektu NBD (15).

Z oddělení špatných zpráv: Ani NBD nepomůže k bezbolestnému (čtěte: nepadajícímu) swapování po síti. K tomu jsou nutné netriviální změny v memory-managementu a ty se rozhodně hned tak nepodaří prosadit…

Pohled do budoucnosti

Času na vývoj bylo bohužel k dispozici pouze konečné množství, takže zdaleka ne všechno, co si kdo přál, bylo dokončeno. Na verzi 2.3 bylo například odloženo:

- Přenos IP přes SCSI (není od věci mít dva počítače spojeny linkou rychlosti 80 MByte/s).
- Přenos IP po Fibre-Channelu (dokonce 1 Gbit/s).
- PLIP využívající rychlé obousměrné paralelní porty.
- Předělání bridgingu (celý bridge group by se ke zbytku jádra měl chovat jako jediný interface).
- Přímý přenos packetů přes DMA z uživatelského prostoru do síťových karet či zpět.
- Integrace podpory ATM a HIPPI.

Závěrem

Tento výčet „síťových novinek“ ukazuje, že jádra 2.1 toho stran síťování opravdu mohou nabídnout mnoho nového. Za zmínku ovšem stojí, že ani kernelům 2.0 se vylepšování nevyhnulo — i tam byly postupně (po důkladném prozkoušení ve 2.1) některé věci z 2.1 přeneseny: například *autofs* či *socket hashing*. ■

1 Linuxový seminář MFF UK
http://sunsite.mff.cuni.cz/linux/seminar/Linux.shtml
2 IP routing commands
http://ftp.gts.cz/FTP/pub/MIRRORS/ftp.inr.ac.ru/ip-routing/
3 Cisco
http://www.cisco.com
4 IP Firewalling Chains
http://www.adelaide.net.au/~rustcorp/ipfwchains/
5 `ipautofw`
ftp://ftp.netis.com/pub/members/rlynch/
6 `ipmasqadm`
http://juanjox.home.ml.org/
7 Port Forwarding
http://www.monmouth.demon.co.uk/ipsubs/portforwarding.html
8 `ipportfw`
ftp://ftp.compsoc.net/users/steve/ipportfw/linux21/
9 Stránka o IPv6
http://playground.sun.com/pub/ipng/html/ipng-main.html
10 Linux IPv6 FAQ/HOWTO
http://www.linuxhq.com/IPv6/index.html
11 Stránka Donalda Beckera o síťových ovladačích
http://cesdis.gsfc.nasa.gov/linux/drivers/
12 The Project List
http://atrey.karlin.mff.cuni.cz/~mj/plist.html
13 CODA Distributed Filesystem
http://www.coda.cs.cmu.edu/
14 X.25 a Frame-Relay síťování s Linuxem
http://users.skynet.be/kribonne/linux-x.25/index_old.html
15 Projekt NBD
http://atrey.karlin.mff.cuni.cz/~pavel/nbd/nbd.html

Báječný svět jádra v. 2.2

přeložil Nathan L. Cutler, 14. srpna 1998

Jak vám zajisté řekne jakýkoliv vývojář jádra, blíží se zveřejnění Linuxu 2.2.

Zatímco Linux 2.1 se pomalu, ale jistě zdokonaluje (přičemž čísla verzí stoupají do závratných výšek — v době, kdy píši tento text, už je verze 2.1.115), všichni netrpělivě očekávají den, kdy verze 2.2.0 bude standardním kernelem jednotlivých distribucí. Verze 2.2 linuxového jádra je důležitým milníkem a je dobré vědět, v čem spočívá její rozdílnost, i když vývoj kernelu „tolik“ nesledujete. Proto vám předkládám tyto postřehy ohledně novinek ve vývoji jádra v poslední době. Týkají se hlavně x86, neboť tento Linux používám doma nejčastěji.

Podotýkám, že tento text nepojednává o všech nových typech hardwaru, které Linux podporuje. Za prvé, mnoho periferií (např. scannery a tiskárny) je obsluhováno výhradně z uživatelského prostoru. Jiné, jako jsou videokarty a myši, jsou ošetřeny kombinací uživatelských a kernelových ovladačů. Pokud v tomto textu nenarazíte na ten či onen kus železa, který vás zajímá, zdaleka to neznamená, že Linux 2.2 ho nepodporuje. Naopak je docela pravděpodobné, že podporuje, ale možná jinými než kernelovými prostředky.

Matoucí spleť procesorů

Je velmi zajímavé sledovat postupný vývoj procesorů firmou Intel — pokud zrovna nemáte nic lepšího na práci. Merced, Celeron, MMX… neustále se objevují nová jména technologií, a neustále je nahrazují nové, ještě moder-

nější technologie. (Otázku, zda a nakolik tyto technologie „stojí za to“, nehodlám v žádném případě rozvíjet.) Firmě Intel již docela solidně konkurují další firmy (AMD, Cyrix ad.), jejichž jednotlivé procesory mají své zvláštnosti, malé optimalizace a chybičky. Je velmi těžké se v tom všem orientovat.

Linux 2.2 bude prvním stabilním jádrem, podporujícím nejnovější možnosti optimalizace v rámci konfigurace před překladem jádra. Linux 2.2 (i pozdější revize kernelu v. 2.0) obsahuje úpravy, které opravují nejrozšířenější chyby procesorů nebo je alespoň neutralizují. Jako příklad ať slouží nechvalně proslulý bug „F00F“ v Pentiu. Jiné chyby, které se nedají na úrovni kernelu neutralizovat, jsou detekovány a ohlášeny během startu.

Merced je zatím v nedohlednu, ale Linux 2.2 již byl portován na Sparc64, Alpha a další 64-bitové platformy, takže infrastruktura 64-bitového jádra již byla „vybudována“. (Samozřejmě existují jiné překážky, které by se musely překonat dříve než by bylo možné zveřejnit Linux/Merced,i ale pouhá existence 64-bitového jádra je významným krokem.)

Víceprocesorové stroje budou mnohem výkonnější pod 2.2 než pod 2.0, neboť problémy jako *global spinlock* byly odstraněny. Podpora se vztahuje až na 16 procesorů (stejně, jako v 2.0), ale i tak by výkonový rozdíl měl být úžasný. Dále je nyní nově podporováno IO-APIC v intelovských boardech, což znamená lepší podporu SMP obecně.

V ostatních portech bude Linux 2.2 obsahovat zlepšenou podporu řady strojů typu server jako jsou Sparc, Sparc64 a Alpha. Co se týče strojů typu desktop, Linux 2.2 již běží na Macintoshích (ve variantách m68k i PPC) s různým stupněm hardwarové podpory. (Dá se předpokládat, že podpora se bude jen zlepšovat až se začne pracovat na další kernelové řadě (2.4?).) Rovněž je snaha portovat Linux na jiné procesory, např. ARM, které jsou čím dál populárnější v tzv. *embedded systémech*.

Netýká se to přímo verze „dva-dva“, ale snad stojí za zmínku, že pokračují práce na podmnožině linuxového jádra pro stroje s procesory 8086, 8088, 80186 a 80286. Nejspíše to v době zveřejnění verze 2.2 nebude hotovo, ale uživatelé těchto strojů se na to mohou těšit v některých z budoucích kernelových release.

Je to trochu smutné, ale množství nových vlastností, které bylo do Linuxu 2.2 přidáno, způsobilo jeho zvětšení tak, že tato kernelová řada zabírá podstatně více paměti než její předchůdkyně. Nezbývá než doufat, že budou do release zahrnuty volitelné položky, které umožní stlačit paměťové nároky na rozumnou míru. Takže skutečnost, že absolutní minimum paměti RAM, nutné k provozu textové linuxové instalace již vzrostlo na 5 Mb, zde uvádím s jistou mírou lítosti. (U Linuxu 2.0 bylo minimum 4 Mb.) Přibližná hranice rozumné použitelnosti bez swapování však zůstává kolem 8 Mb. Na druhé straně, pokud máte to štěstí mít počítač vybavený větší pamětí, můžete si být jisti, že Linux 2.2 tuto skutečnost využije více, než kdykoliv jindy, má totiž několik „vymakaných“ cacheovacích režimů a jiné optimalizace, s kterými dostane z vašeho železa maximální výkon. ■

Jak na Linuxdoc-sgml

Michal Choura, 2. srpna 1998

Co to je sgml

Balík programů `sgml-tools` byl vytvořen Mattem Welshem jako jednotné prostředí pro psaní dokumentů projektu LDP (Linux Documentation Project). Utility jsou nadstavbou nad SGML a autoři mohou psát své texty v jednoduchém a přehledném formátu, jakým SGML je. Dokument je potom možné zformátovat do podoby čistého textu, systému info, HTML stránky s hypertextovými odkazy, LaTeXu či Postscriptu k vytištění, a dokonce i do formátů editoru LyX nebo do rtf. Struktura SGML dovoluje jednoduše napsat další konvertory do jiných, třeba i dnes neznámých formátů.

Tento článek popisuje versi 1.0.3 balíku `sgml-tools`. Starší verse dodávané např. s distribucí Linuxu Red Hat 5.0 nemají preprocesor a mohou se odlišovat i v jiných vlastnostech od tohoto textu.

Základní struktura dokumentu

Každý dokument v Linuxdoc-sgml má takovouto základní kostru:

```
<!doctype linuxdoc system>

<article>

<title>Co z toho asi bude
<author>Bílá Paní

<date>31. prosince 1999

<abstract>
Informace o generálním sjezdu všech strašidel
   k problematice roku 2000.
</abstract>

<toc>

</article>
```

Formátování textu určují příkazy, uvedené v úhlových závorkách <>. Většina příkazů se vztahuje jen na blok textu, potom je ohraničena znaky <příkaz> a </příkaz>.

Každý SGML dokument musí začínat řádkem `<!doctype linuxdoc system>`. Ten specifikuje DTD typ SGML dokumentu. Příkaz `<article> ... </article>` říká, že použijeme styl „article“. Význam značek `<title>`, `<author>` a `<date>` je zřejmý, `<abstract> ... </abstract>` může obsahovat stručný popis dokumentu. Příkaz `<toc>` vysází ve výsledném textu obsah.

Nadpisy kapitol, odstavce

Pro nadpisy kapitol se používají příkazy `<sect>`, `<sect1>`, `<sect2>`, `<sect3>` a `<sect4>`. Syntaxe v dokumentu je následující:

```
<sect>Kapitola

<sect1>Podkapitola

<p>
Text odstavce.
```

Za zmínku stojí především uvedení značky `<p>`. Mezi nadpisem kapitoly a textem být musí, jinak by konvertor nepoznal, kde končí nadpis a začíná odstavec. Při oddělo-

vání odstavců mezi sebou je možné použít buď <p>, nebo stačí jen vynechat prázdný řádek.

Zvýrazňování textu

Základní způsob zvýraznění textu je kursiva. Tu vysadíme příkazy ...kursiva... . Sazbu tučného textu je možno dosáhnout příkazy <bf> ...tučně.. </bf>. Neproporcionální font psacího stroje lze vytvořit příkazy <tt> ...strojově... </tt>.

U podobných příkazů je povolena ještě syntax tohoto tvaru: <em/ ... /. Pochopitelně, že text uvnitř příkazu nesmí obsahovat lomítka. To připouští pouze první syntax.

Prostředí verbatim pro sazbu textu beze změny tak, jak je ve zdrojovém textu, lze vytvořit značkami

```
<verb>
Nějaký text. @#$%^&.
</verb>
```

Uvnitř verbatim prostředí musí být použit příkaz &ero; pro znak & a &etago; místo dvojice znaků </. Uvnitř verbatim prostředí se nesmí vyskytovat ani sekvence \end{verbatim}, protože koliduje s LaTeXovým koncem verbatim prostředí.

Obdobný příkazu verbatim je <code> main(){} </code>. Ten se užívá k sazbě zdrojového kódu. Jediný rozdíl oproti verb je, že před a za textem v příkazu se umístí vodorovná čára přes celou šířku sazby.

Prostředí tscreen nastaví font na tt a odsadí sázený text. Ten je opět zadán ve tvaru

```
<tscreen>
Nějaký text.
</tscreen>
```

Obdobné prostředí quote provede totéž, ale nenastaví font na tt. Užívá se pro sazbu citátů.

Výčty

Linuxdoc-sgml umí sázet tři druhy výčtů:

- itemize jsou položky označené puntíkem,
- enum jsou automaticky číslované a
- descrip je popisný výčet.

První dva typy výčtů mají tuto strukturu:

```
<itemize>
  <item>První položka,
  <item>Druhá položka.
</itemize>
```

Jak výčet dopadne je vidět na začátku odstavce. Poslední, popisný výčet, se vytváří trochu jinak:

```
<descrip>
  <tag>Položka</tag> A její vysvětlení.
  <tag>Jiná</tag> Položka jiná.
</descrip>
```

Sazba speciálních znaků

V SGML je mnoho tzv. speciálních znaků, které je třeba sázet sekvencemi tvaru &něco;. Mezi nejdůležitější patří:

- & — &
- < — <
- > — >
- $ — $
- # — #
- % — %
- ~ — ˜
- " — &dquot;
- | — |
- × — ×
- @ — @

Takových znaků je podstatně více, doporučuji podívat se do dokumentace, dodávané s balíkem sgml-tools.

Odkazy a reference

Pokud se chcete odkazovat na nějakou kapitolu, je třeba přidat za nadpis kapitoly takovouto definici:

```
<sect>O bezhlavém rytíři<label id="hlava">
```

V textu je potom možné se odkazovat na tuto kapitolu příkazem <ref id="hlava" name="O bezhlavém rytíři">. Argument name se využije při tvorbě HTML verse dokumentu, kde nebude uvedeno číslo, ale přímo název kapitoly s hypertextovým odkazem na ni. V názvech odkazů, tedy v argumentu id, by se neměly vyskytovat žádné speciální znaky, ani podtržítko _, ani české znaky.

Odkazy do prostoru Internetu mají formát:

```
<url url="http://www.strasidla.org/~jozin/"
   name="Jožin z bážin">
```

Argument name se použije v HTML versi jako odkaz a v TeXové versi pro popis odkazu.

Pro odkazy na emailové adresy je lepší použít alternativu

```
<htmlurl url="mailto:hejkal@huste-lesy.edu"
name="mailto:hejkal@huste-lesy.edu">
```

Tak se při výstupu do textu neobjeví adresa dvakrát za sebou, jednou jako URL a podruhé jako popis, ale přesto zůstane správný odkaz v HTML versi.

Podmínky „preprocesoru“

Novější verse sgml-tools obsahují velmi jednoduchý preprocesor. S jeho použitím je možné vytvářet odlišný text pro různé výstupní formáty. Základní struktura je:

```
Tento <#if output="latex">TeXový</#if> dokument...
```

Slovo TeXový se objeví jen v LaTeXové versi dokumentu. V příkazu je povolen i znak „nebo“: name="latex|html" vysází takový podmíněný text v LaTeXové i HTML versi.

Jako argument příkazu #if je možné použít i libovolnou konstantu, třeba version="oficialni". Pokud potom version bude nastaveno na oficialni nebo nebude nastaveno, výstup bude text v podmínce obsahovat. Konstanta se nastaví na příkazové řádce při překladu dokumentu (při výstupu do HTML) takto:

```
$ sgml2html -D version=moje text.sgml
```

Obdobně existuje příkaz <#unless>, který se chová opačně než #if. Tedy vloží podmíněný text jen tehdy, neplatí-li podmínka uvedená v argumentu.

Na tyto příkazy Linuxdoc-sgml je třeba dát si obzvláště pozor. Protože jsou zpracovány preprocesorem, může se při chybném použití stát, že samotný překladač SGML bude vydávat zcela nesrozumitelné chybové hlášky na úplně jiných místech textu.

Poznámky k české sazbě

V češtině je pravidly pravopisu stanoveno, že by neměly na koncích řádků zůstávat jednoslabičné předložky. V SGML je na takovou možnost pamatováno a je definována, stejně jako v TEXu, pevná mezera, ve které nikdy nebude proveden řádkový zlom. Tato mezera se zapisuje jako vlnovka ~ a měla by být ve všech ke zlomu nevhodných místech uváděna.

Také uvozovky se v češtině zapisují jinak než v jiných jazycích. Doporučuji i v SGML používat pro počáteční uvozovky dva znaky čárka hned za sebou (,,) a pro koncové dva znaky zpětný apostrof za sebou (‘‘). TEX v takovém případě vysází správné české uvozovky a i v ostatních formátech to vypadá lépe než znak palec (").

Překlad Linuxdoc-sgml

Balík sgml-tools obsahuje sadu skriptů, které poslouží k pohodlnému překladu dokumentů. Nejdůležitější z nich jsou:

- sgmlcheck strasidla.sgml — Provede syntaktickou kontrolu dokumentu.
- sgml2txt -c latin strasidla.sgml — Vygeneruje textovou versi dokumentu. Je potřeba říci, že dokument je v latin 1, jinak poničí české znaky.
- sgml2html strasidla.sgml — Vygeneruje stránky HTML jazyka.
- sgml2latex -p a4 -o tex strasidla.sgml — Vygeneruje LATEXový zdrojový kód. S touto versí je pro češtinu nejvíce práce. Protože sgml-tools češtinu neznají, musí se vygenerovat tímto příkazem zdrojový kód pro cslatex a na jeho začátku nahradit řádku \usepackage[latin1]{inputenc} za řádek \usepackage{czech}. Tím načteme styly cslatexu a budeme mít správně dokonce i dělení slov. Pak se nejméně dvojím překladem cslatex strasidla vygeneruje DVI a případně dvips strasidla i Postscript.

Další konvertory jsou sgml2info, sgml2lyx a sgml2rtf.

Kde lze sgml-tools získat a závěrečné postřehy

Balík sgml-tools má svou domovskou stránku (1), na které lze získat nejnovější versi a i další podrobnější dokumentaci.

Linuxdoc-sgml je užitečný balík pro psaní dokumentace, u které je požadován výstup do co největšího počtu formátů spíše, než vytříbená typografická kvalita. Tištěné verse dokumentů mají daleko ke kráse textů psaných přímo v TEXu. Přesto je na příkladu celého Linux Documentation Projectu vidět, že jej lze použít i na velkých projektech s dobrými výsledky. Jako ostatně vždy, musí český uživatel počítat s tím, že narazí na drobné problémy v důsledku používání svého rodného jazyka. ■

1 Sgml-tools
http://www.sgml-tools.org/

Začínáme s Emacsem: I — Spuštění a ukončení

Michal Fadljevič, 11. září 1998

Vítám vás u prvního dílu seriálu o Emacsu. Chtěl bych v něm přístupnou formou naučit člověka, který nemá (zatím) v Unixu svůj oblíbený editor, jak s Emacsem žít a přežít. Nepovažuji se za nějakého emacsového guru a doufám, že právě proto by mohl být seriál srozumitelný.

Proč jsem vybral Emacs

Když jsem před lety začal pracovat v Unixu, začal jsem hledat editor, který by splňoval moje požadavky:

- velmi mocný (a konfigurovatelný)
- běžící všude (Unix, DOS, Win*)

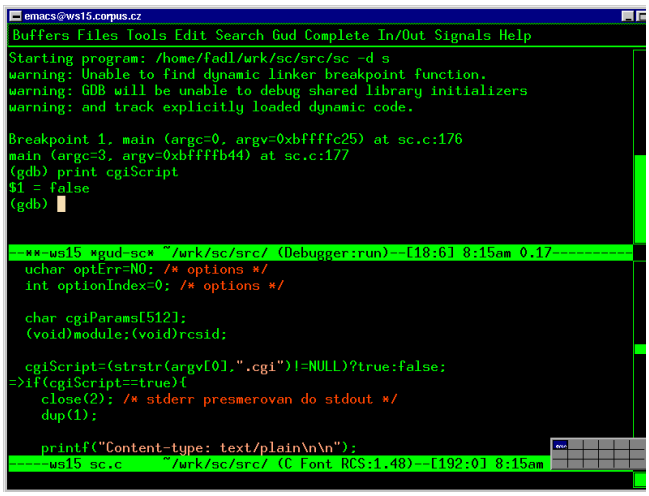
Emacs oba tyto požadavky splňuje, takže jsem spokojený člověk :-).

Co se v Emacsu dá dělat

Je dobré říci si hned na začátku to, co vždycky s humorem říkají Emacsoví guru, a sice, že:

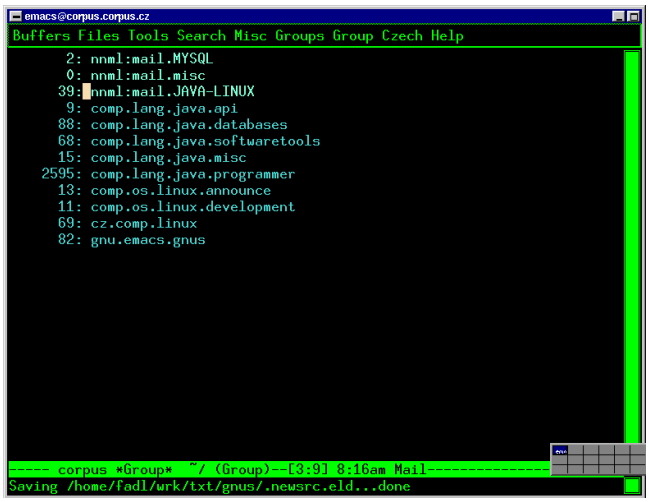
> V Emacsu je možné všechno.

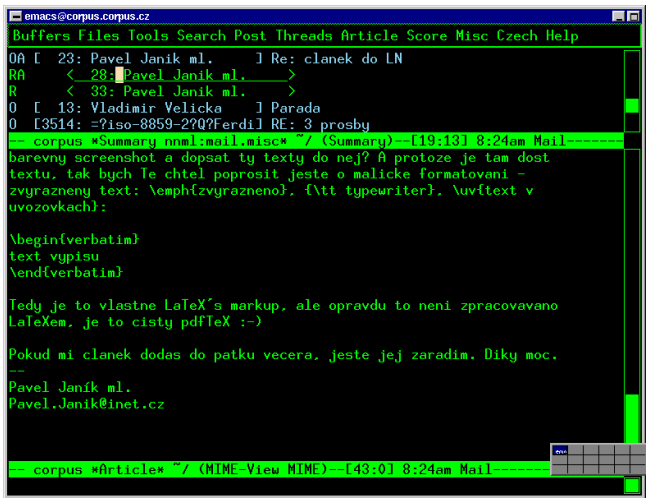
Skutečně, v Emacsu můžete s velkým komfortem psát knihy v TEXu, vyvíjet a ladit programy v C/C++/Javě a spoustě jiných jazyků (Emacs si třeba pod sebou spustí gdb nebo dbx a vy na jedno stisknutí klávesy provádíte „Next“, „Step“, „Finish“ a Emacs vám zobrazuje pěkně graficky



a sám přepíná zdrojové soubory tak, jak krokujete — prostě ráj programátora), dál můžete číst poštu a newsové skupiny

(další obrázek), editovat binární soubory, počítat si daňové přiznání, vést jednoduché účetnictví, procházet se po anonymních FTP serverech stejně pohodlně jako po lokálních discích (velmi podobně jako v oblíbeném mc) a v podstatě cokoliv, na co seženete příslušný balík maker nebo na co si makra napíšete sami. Ke všem výše zmíněným věcem se dostaneme v průběhu času, jen co budeme mít za sebou úplné začátky. Za sebe ještě na úplný konec úvodu dodávám: „Buďte na svůj editor nároční“, čili všechno co děláte opakovaně si zautomatizujte. V Emascu to skutečně jde.





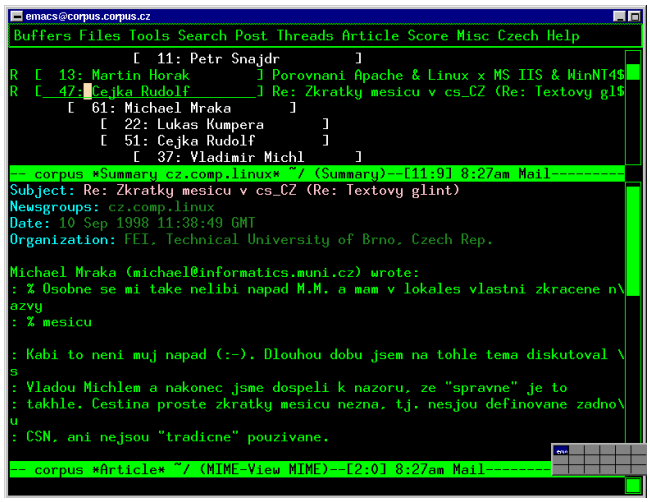
Instalace

Protože instalace Emacsu pro danou distribuci je trochu nad rámec tohoto povídání, budu předpokládat, že používáte Red Hat 4.2 a máte nainstalovány příslušné RPM balíky s Emacsem jako já. Jsou to tyto balíky:

```
$ rpm -qa|grep emacs
emacs-19.34-4
emacs-el-19.34-4
a jeden z balíků
emacs-nox-19.34-4
emacs-X11-19.34-4
```

Pokud máte jinou distribuci, budete si muset příslušné instalační balíky někde iniciativně sehnat, případně popátrejte po standardních GNU FTP mirrorech a tam se dá Emacs najít i ve zdrojových souborech. Používá `autoconf`, takže na většině rozumných systémů vystačíte s obvyklým:

```
$ tar xfvz ema*tar.gz; cd ema*
$ ./configure
$ make
... čas na čaj - někdy i na dva {:)
$ make install
```



Podle mých instrukcí jste zvídaví a hned předchozí zmínka o třech balících vás jistě zaskočila. Takže malé vysvětlení

- `emacs-19.34-4` — zde je vlastní distribuce Emacsu
- `emacs-el-19.34-4` — zde jsou různé další balíky ve zdrojové formě pro zvídavé nebo pro ty, kdo si chtějí třeba opravit nějakou chybu
- `emacs-X11-19.34-4` — `/usr/bin/emacs`
- `emacs-nox-19.34-4` — `/usr/bin/emacs-nox`

Pokud chcete spouštět Emacs pouze v čistě znakovém režimu, stačí vám balík `emacs-nox` a obdobně jste-li zarytý „X Window-sák“, nainstalujete si zřejmě `emacs-X11`. Rozhodně vám stačí jen jeden z nich.

```
/usr/bin/emacs ... 1908308 B
/usr/bin/emacs-nox ... 1473692 B
```

Rozdíl je mezi nimi (kromě velikosti) ten, že verze zkompilovaná pro X Window System používá přímo menu v horní části obrazovky, posuvnou lištu po pravé straně a při běhu některé dialogy vyskakují jako nová dialogová okna, zatímco verze `-nox` tyto vymoženosti nemá. Jinak funkcionalita zůstává stejná. To je vlastně jedna z hlavních myšlenek tohoto hlavního proudu tzv. *GNU Emacsu* a sice, že přenositelnost by měla zůstat maximální. Jen pro vysvětlení, během let vznikaly různé názorové proudy a skupiny vývojářů Emacsu se svobodně štěpily a štěpily, takže dodnes můžete najít (a případně používat) *XEmacs*, *Lucid Emacs* atd., a je jich ještě více. Jak asi správně tušíte, XEmacs se chce více integrovat do X Window Systému a zřejmě mu nevadí, že pak už nepůjde spustit na ubohé konzoli nebo přes terminálový emulátor po modemu. My se budeme věnovat pouze GNU Emacsu, tzn. tomu Emacsu, který skutečně běží všude (řádkový režim, X Window System, MS-Windows,

DOS) a tím, že jste ho pustili třeba na konzoli, neztrácíte žádnou z jeho úžasných schopností.

Pokud vám velikost nevadí, nainstalujte si `emacs-X11`. Pokud ho spustíte v X Window System např. z `xtermu`, naskočí vám normálně nové okno s Emacsem a ve znakovém režimu ho udržíte tak, že ho spustíte:

```
$ emacs -nw
```

Jako: spusť Emacs bez okna (`-nw` „NoWindow“). Pak nevyskočí nové okno, ale pěkně se rozběhne spořádaně v okně, kde jste napsali tenhle řádek.

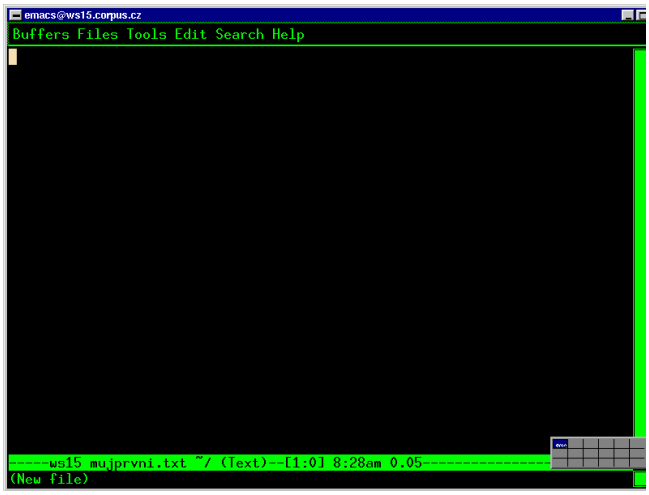
Spuštění

Emacs se dá použít běžným způsobem jako `vi`, `joe`, `vim`, `elvis`, `pico` apod. Postupem času dělá člověk stále více věcí přímo v Emacsu, takže správný Emacs-lover spouští Emacs jen ráno a vypíná ho až večer.

```
$ emacs mujprvni.txt
```

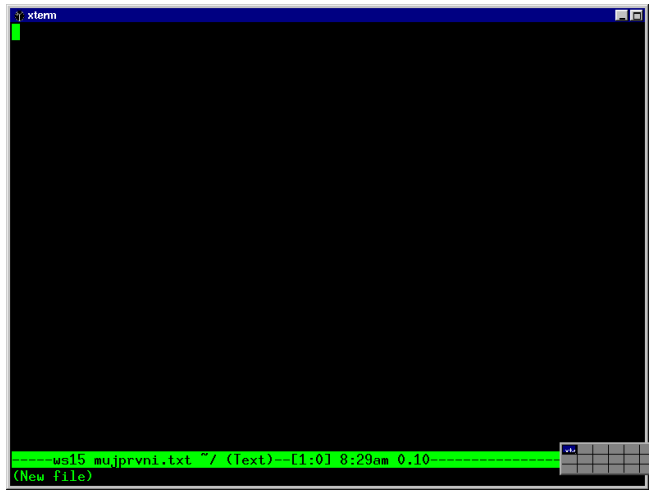
resp. na konzoli

```
$ emacs -nw mujprvni.txt
```



Ocitli jste se ve světě Emacsu. Všechno zde má svá pravidla a přiznávám otevřeně, že to chce určité úsilí a čas zvyknout si. (PS: První dva měsíce jsou nejhorší, pak je to zase naopak úžasné). Na obrázku vidíte (v pořadí shora dolů) *menu*, velkou část obrazovky (*pracovní oblast*) sloužící pro vlastní práci, vpravo *posuvnou lištu* (vyzkoušejte si pravé, levé a prostřední tlačítko), v dolní části pak velmi užitečný (a samozřejmě konfigurovatelný) *stavový řádek* (mode-line) a úplně poslední řádek je snad nejdůležitější část a sice *komunikační oblast* (minibuffer), kde se odehrává velká část Vaší komunikace s Emacsem. Emacs vám tam zobrazuje různé varování, chyby, rady a zároveň tam odpovídáte jednoduché otázky apod. Když už jsme u těch otázek. Všimněte si, že u „méně důležitých“ otázek vystačí s odpovědí `y` / `n`, ale u „destruktivnějších“ vyžaduje pěkně vypsat `yes`/`no` — to proto, aby vás donutil uvědomit si, jestli opravdu chcete danou věc udělat. Šikovné, nemyslíte? Dívejte se tam často. Emacs vám tam často radí, co můžete dělat a jak se dočíst nápovědu.

Obrázek ukazuje, jak vypadá Emacs při práci ve znakovém režimu. Přicházíte pouze o menu v horní části. To je upřímně pro začátečníky docela praktická věc, protože u položek jsou pěkně přehledně připsány klávesové sekvence, které akci vyvolají. Během několika měsíců si tak člověk ty nejčastější nenásilně zapamatuje. Takže pokud máte možnost a slušný výpočetní výkon, pouštějte si (ze začátku) raději X-ovou verzi.



Platí to, že Emacs s vámi neustále komunikuje, t.j. měli byste vidět nebo slyšet odezvu na to co děláte. Pokud se tak neděje, je to divné. V takových situacích mačkejte nejprve jednou a někdy i vícekrát po sobě `C-g`, což je ve všech situacích stornující kombinace kláves, která přeruší a skončí to, co jste právě dělali (a zřejmě se vám to z nějakého důvodu nedaří).

Teď trochu typografické konvence. V Emacsu je hodně povelů a často se používají klávesové sekvence a to někdy s přepínači a někdy bez přepínačů. Takže

zápis	znamená
`C-g`	přidrž přepínač `Ctrl` — `Control` a zmáčkni jednou písmeno `g`
`C-g` `C-g`	přidrž přepínač `Ctrl` a zmáčkni dvakrát písmeno `g`
`C-g` `g`	přidrž `Ctrl` a zmáčkni jednou písmeno `g` potom pusť `Ctrl` a zmáčkni ještě jednou písmeno `g`
`M-v`	zmáčkni přepínač `Meta` — u nás většinou `Alt` — a jednou písmeno `v` Pokud toto nefunguje, používá se místo `M-v` náhrada `ESC` `v`, čili zmáčkni `ESC` a potom písmeno `v`
`C-M-v`	přidrž současně `Ctrl` a `Meta` a jednou písmeno `v`. Zde se nebojte — pokud vám z nějakého důvodu nefunguje `Meta`, ekvivalent je skutečně `Esc` `C-v`
`C-x` `r` `l`	Nejprve `C-x`, uvolnit všechno a postupně klávesy `r` a `l`
`C-x` `s`	Nejprve přidrž `Ctrl` a písmeno `x`, pak uvolni `Ctrl` a zmáčkni `s`
`C-x` `C-s`	Nejprve přidrž `Ctrl` a písmeno `x`, a pak zmáčkni `s`

Zde si všimněte důležitého rozdílu — v posledním přípa-

dě pořád držíte Ctrl stisknutý, zatímco v předposledním ho po písmenu x uvolníte. Na to si brzy zvyknete, jen je potřeba si dát ze začátku trochu pozor. Tak a teď už jsme definovali konvenci zápisu, takže teď už víte, co znamená, když vám prozradím nejnejdůležitější věc a sice — jak skončit s Emacsem — je to pomocí sekvence C-x C-c.

Než se dostaneme k tomu, co která sekvence dělá, malé vysvětlení toho, proč toto asi vzniklo. Celkem jednoduchou úvahou dostáváme:

Má-li program psát text, musí většina kláves po stisku vložit do textu příslušný znak. To ano, ale musíme se nějak textem pohybovat, opravovat ho a podobně. Dobrá, můžeme použít přepínač + klávesa např. C-a, C-b, až C-z nebo M-a, M-b až M-z. Jak časem uvidíte, příkazů umí Emacs na stovky, takže ani to by nestačilo, proto se používá i C-M-a až C-M-z a navíc různě dlouhé sekvence po prefixech C-x a C-c. A všechny tyto sekvence jsou vámi modifikovatelné. Čili ad absurdum po spuštění Emacsu jenom Vy můžete vědět, co jste si posledně přiřadil za akci třeba s sekvenci C-c v v d r. No, to už je opravdu trochu zvrhlé…

klávesa	nebo	akce
→	C-f	znak vpřed
←	C-b	znak vzad
↑	C-p	řádek nahoru
↓	C-n	řádek dolů
Del	C-d	smaž znak pod kurzorem
BackSpace		smaž znak vlevo
	C-k	smaž řádek
Home	C-a	jdi na začátek řádku
End	C-e	jdi na konec řádku
PgUp	M-v	stránka nahoru
PgDn	C-v	stránka dolů
	C-x C-s	ulož tento soubor
	C-x C-c	ukonči Emacs

Řekněme si typickou konfiguraci. Po instalaci máte (poměrně) rozumné vazby nadefinovány a měnit budete jen to, co vám bude opakovaně vadit a hlavní, co budete dělat je, že si budete přidávat svoje nové vazby (*bindings*). Budu se snažit psát do závorek anglické ekvivalenty proto, abyste měli lepší vazbu na to, o čem se píše v dokumentaci. (Do ní budete muset dříve či později stejně vklouznout, protože tento seriál vám nemůže objasnit vše. Jen pro zajímavost, úplná referenční příručka Emacsu má něco kolem 600 stran — taková bible Emacsu).

Takže ještě jednou, pokud řeknu, že C-x C-s uloží soubor, který právě editujte, znamená to přesně to, že ve standardní distribuci Emacsu po prvním rozběhnutí je na sekvenci C-x C-s přivázána funkce „ulož soubor“. Pokud jste něco měnili, je to na vás. Čili opatrně na změny, které děláte v konfiguraci Emacsu. Z 90% se pak člověk rozčiluje neprávem na Emacs, spíše by měl spílat sám sobě. (Mluvím z vlastní zkušenosti.) Všechny takovéto úpravy si člověk typicky přidává do svého souboru `~/.emacs`, takže pokud byste si ho nějak pokazili, jednoduchá pomoc — smažte ho a Emacs se rozběhne v defaultní konfiguraci, jak byl sestaven.

Protože Emacs je určen po práci v mnoha různých podmínkách, není zvyklý se na cokoliv vázat. V podstatě jakákoliv klávesnice bude dobrá. I když třeba nebude mít F1 až F12 nebo klávesu Alt a podobně. To neznamená, že nemáte používat klávesy End, Home, PgUp, PgDn, F1 až F12, Ins a Del. Většina z nich funguje, jak je obvyklé, a těm ostatním to vysvětlíme v některém z příštích dílů. Je dokonce velmi pravděpodobné, že vám teď hned třeba např. Del nebude dělat to, co má — to je nejčastější problém, ale na všechno se podíváme a v tabulce vidíte, že Del má ekvivalent C-d, takže se tak moc neděje…

Nyní si zkuste napsat nějaký odstavec textu a vyzkoušejte si editování.

Není to tak hrozné, co? Zkuste si jako samostatné cvičení „oeditovat“ několik souborů. — To by bylo pro dnešek vše. Pokračování příště. Pokud vám bylo něco nejasné nebo máte nápad, co byste se chtěli dočíst o Emacsu příště, napište mi. ■

Linux a vlákna

Vladimír Michl, 7. srpna 1998

Tento článek si klade za úkol seznámit čtenáře s vlákny v Linuxu a jejich použitím, případně s tím, čeho by se měl člověk při používání vláken vyvarovat.

Ale od začátku. Nejprve je třeba osvětlit rozdíl mezi termínem proces a vlákno.

Jako proces je v systému chápán souhrn kódu programu, dat programu, zásobníku, údajů o procesem otevřených souborech, a také informací ohledně zpracování signálů. Tyto všechny informace má každý proces vlastní (privátní) a nemůže je sdílet s jiným procesem, kromě datových oblastí. Při volání jádra `fork(2)` se pak tyto informace pro nový proces skopírují, takže jsou pro něj zase privátní.

Jako vlákno si můžeme představit odlehčený proces, tj. pouze kód vlákna a zásobník, vše ostatní je sdíleno s ostatními vlákny téhož procesu. Vlákno je tedy podmnožinou procesu a proces může vlastnit několik vláken. Vlákno samo o sobě v systému existovat nemůže, musí k němu vždy existovat proces, se kterým sdílí všechna data, otevřené soubory, zpracování signálů.

Pro implementaci vláken existují tyto modely:

- *one-to-one* — Implementace provedena na úrovni jádra. Každé vlákno je pro jádro samostatný proces, plánovač procesů nečiní rozdíl mezi vláknem a procesem. Nevýhodou tohoto modelu může být velká režie při přepínání vláken.
- *many-to-one* — Implementace provedena na úrovni uživatele, program si sám implementuje vlákna a vše okolo. Jádro o vláknech v procesech nemá ani tušení. Tento model se nehodí na víceprocesorové systémy, protože vlákna nemohou běžet zároveň (každé na jiném procesoru), jeden proces nelze nechat vykonávat na dvou procesorech. Výhodou může být malá režie přepínání vláken.
- *many-to-many* — Implementace provedena na úrovni jádra i uživatele. Tento model eliminuje nevýhody předchozích implementací (velká režie při přepínání proce-

```
#include <pthread.h>
#include <stdio.h>
#define ITEMS 10000

void * process(void *a){
      int i;
      printf("Process %s: start\n", (char *)a);
      for (i = 0; i<ITEMS; i++){
            printf("%s", (char *)a);
      };
      printf("Process %s: end\n", (char *)a);
      return NULL;
}

int main(){
      int retcode;
      pthread_t a,b;
      void * retval;

      retcode = pthread_create(&a, NULL, process, "A");
      if (retcode != 0) fprintf(stderr, "create a~failed %d\n", retcode);
      retcode = pthread_create(&b, NULL, process, "B");
      if (retcode != 0) fprintf(stderr, "create b failed %d\n", retcode);
      retcode = pthread_join(a, &retval);
      if (retcode != 0) fprintf(stderr, "join a~failed %d\n", retcode);
      retcode = pthread_join(b, &retval);
      if (retcode != 0) fprintf(stderr, "join b failed %d\n", retcode);
      return 0;
}
```

Výpis č. 1: Příklad použití vláken

sů, souběžně nemůže běžet více vláken) a je proto použit v mnoha komerčních UNIXech (Solaris, Digital Unix, IRIX).

V Linuxu je použit model první. Nevýhoda velké režie v podstatě není, protože přepínání procesů je v Linuxu implementováno velmi efektivně. Pro tvorbu procesů a vláken se v Linuxu používá volání jádra `clone(2)`, které ale používají pouze knihovny obhospodařující vlákna.

V začátcích, kdy se v Unixech začala vlákna objevovat, měl každý unixový operační systém jiné aplikační rozhraní pro práci s vlákny, a proto byly programy špatně přenositelné. Proto vznikla norma POSIX, která mimo jiné také definuje aplikační rozhraní pro práci s vlákny (POSIX 1003.1c). Toto POSIXové rozhraní je dostupné i na OS Solaris 2.5, Digital Unix 4.0, IRIX 6. S každou distribucí Linuxu postavenou na `glibc-2.0` je dodávána knihovna `pthread`, která právě toto POSIXové aplikační rozhraní implementuje.

Tvorba vláken a jejich ukončení

Pro vytvoření a ukončení vlákna lze použít následující funkce:

```
int  pthread_create(pthread_t * thread,
 pthread_attr_t * attr,
 void * (*start_routine)(void *), void * arg);
```

- funkce vytvoří nové vlákno, které bude vykonávat funkci `start_routine`, což je funkce akceptující jeden parametr typu `void *`. Na adresu `thread` je uložen identifikátor vlákna a jako atributy vlákna můžeme uvést `NULL` pro implicitní hodnoty.

```
void pthread_exit(void *retval);
```

- tato funkce předčasně ukončí vlákno, ze kterého byla funkce zavolána.

Vlákno se také ukončí, skončí-li funkce `start_routine`. V obou případech se předává návratový kód.

```
int pthread_join(pthread_t th,
  void **thread_return);
```

- funkce čeká na ukončení vlákna `th`. Na adresu `thread_return` je uložen návratový kód vlákna.

Příklad, jak funkce použít, naleznete na výpise Příklad použití vláken.

Překlad programu

Při překládání programu, který používá vlákna, je třeba tomuto přizpůsobit hlavičkové soubory knihovny `glibc` tak, aby byly reentrantní. To provedeme definováním makra `_REENTRANT`. Dále je třeba program slinkovat s knihovnou `pthread`. Pro překlad programu na výpise Příklad použití vláken použijeme:

```
gcc -D_REENTRANT -o example1 example1.c -lpthread
```

Kritické sekce pomocí mutexu

Nejprve si položme otázku, co je to kritická sekce. Za kritickou sekci považujeme tu část kódu vlákna, která operuje nad sdílenými daty a hrozí, že paralelně může jiné vlákno

operovat nad stejnými daty. Důsledkem může být nekonzistence dat. Například jedno vlákno zvýší sdílenou proměnnou A o jedna a dále s ní počítá, kdežto druhé vlákno proměnou A zmenší o dvě a dále s ní počítá. Pokud se poštěstí, tak se instrukce mohou proložit tak, že ani jedno vlákno nedá správný výsledek. Tomuto je třeba zabránit a to tím, že do té části, která pracuje s proměnnou A může vstoupit pouze jedno vlákno, druhé musí čekat až to první skončí. Takovéto kritické sekce, kde může být v jednom okamžiku pouze jedno vlákno, nazýváme MUTEX (*MUTual EXclusion*). Mutex má dva stavy — zamčený (*locked* — některé vlákno je uvnitř) a odemčený (*unlocked* — v mutexu nikdo není).

Pro práci s mutexy použijeme funkce:

```
int pthread_mutex_init(pthread_mutex_t *mutex,
       const  pthread_mutexattr_t *mutexattr);
```

- inicializace mutexu

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

- zamčení mutexu. Po návratu je mutex vždy zamčen pro vlákno, které tuto funkci vykonalo. Pokud je mutex již zamčen, funkce pozastaví vlákno a čeká na odemčení mutexu, aby následně mutex zamkla a mohla nechat vlákno pokračovat.

```
int pthread_mutex_trylock(pthread_mutex_t *mutex);
```

- pokus o zamčení mutexu. Pokud je mutex již zamčen, funkce se vrátí s chybou EBUSY.

```
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

- odemčení mutexu

```
int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

- uvolnění zdrojů spojených s mutexem

V příkladu Schematické znázornění použití mutexu můžete vidět použití mutexu.

```
pthread_mutex_t mut_var;
...
/* Inicializace mutexu */
      pthread_mutex_init(&mut_var, NULL);
...
/* Vstup do mutexu */
      pthread_mutex_lock(&mut_var);
/* Vykonání operací nad sdílenými daty */
...
/* Výstup z mutexu */
      pthread_mutex_unlock(&mut_var);
...
/* Na konci programu zrušení mutexu */
      pthread_mutex_destroy(&mut_var);
```

Výpis č. 2: Schematické znázornění použití mutexu

U mutexů se můžeme setkat s tím, že bude třeba mutex zamknout v závislosti na podmínce. Například problém producent – konzument. Producent produkuje data do sdílené proměnné a konzument je čte. Přitom proměnná musí být zabezpečena mutexem a zároveň se musí hlídat stav, zda proměnná obsahuje užitečná data. I na toto POSIX myslí, a to pomocí následujících funkcí:

```
int pthread_cond_init(pthread_cond_t *cond,
       pthread_condattr_t *cond_attr);
```

- inicializace podmínky

```
int pthread_cond_signal(pthread_cond_t *cond);
```

- způsobí spuštění jednoho vlákna, které čeká na podmínce. Jestliže nečeká žádné vlákno, funkce nemá žádný efekt. Čeká-li více vláken, spustí se pouze jedno, ale není definováno jaké.

```
int pthread_cond_broadcast(pthread_cond_t *cond);
```

- způsobí spuštění všech vláken čekajících na podmínce. Jestliže nečeká žádné vlákno, funkce nemá žádný efekt.

```
int pthread_cond_wait(pthread_cond_t *cond,
       pthread_mutex_t *mutex);
```

- automaticky odemkne mutex, pozastaví vlákno a čeká na signál od podmínky. Po příchodu signálu je mutex uzamčen a tato funkce ukončena. Každá podmínka musí být uzavřena v mutexu.

```
int pthread_cond_timedwait(pthread_cond_t *cond,
       pthread_mutex_t *mutex,
       const struct timespec *abstime);
```

- je podobné `pthread_cond_wait()` s tím rozdílem, že čekání je časově omezeno. Pokud čas vyprší, pak je sekce uzamčena a funkce je ukončena s chybou ETIMEDOUT.

```
int pthread_cond_destroy(pthread_cond_t *cond);
```

- uvolní zdroje spojené s podmínkou

V příkladu Použití mutexu v problému producent – konzument můžete vidět použití mutexu a podmínek na problému producent – konzument. Všimněte si rozdílné inicializace podmínek, samozřejmě obě podmínky jdou inicializovat stejným způsobem.

Kritické sekce pomocí semaforů

Semafory se používají pro podobný účel jako mutexy, a to pro kontrolování vstupu do kritických sekcí. Ale na rozdíl od mutexu, kdy v sekci může být pouze jeden, se semafory lze docílit, že v sekci může být více vláken. Semafor si můžeme představit jako počítadlo s počáteční hodnotou, kterou nastaví uživatel. Vždy při vstupu do kritické sekce se čeká, dokud není hodnota semaforu větší než nula. Pokud je, pak se hodnota zmenší o jednu a vstoupí se do kritické sekce. Na konci sekce se hodnota semaforu o jedničku zvedne. Pro práci se semafory používáme funkce:

```
int sem_init(sem_t *sem, int pshared,
       unsigned int value);
```

- inicializace semaforu. Argument `pshared` určuje, zda je semafor lokální pro tento proces (hodnota 0) nebo

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define ITEMS 100
#define NONVALID 0
#define VALID 1
pthread_mutex_t mut_var;
pthread_cond_t condvalid;
pthread_cond_t condnonvalid = PTHREAD_COND_INITIALIZER;
int valid;
int share;
void * konzument(void *a){
      printf("Process %s: start\n", (char *)a);
      while(1){
            pthread_mutex_lock(&mut_var);
            if (!valid)
              pthread_cond_wait(&condvalid, &mut_var);
            valid = NONVALID;
            printf("Process %s: %i\n", (char *)a, share);
            if (share == -1){
                  pthread_mutex_unlock(&mut_var);
                  break;
            };
            pthread_cond_signal(&condnonvalid);
            pthread_mutex_unlock(&mut_var);
      };
      printf("Process %s: end\n", (char *)a);
      return NULL;
}

void * producent(void *a){
      int i;
      printf("Process %s: start\n", (char *)a);
      for (i = 0; i<ITEMS; i++){
            pthread_mutex_lock(&mut_var);
            if (valid)
              pthread_cond_wait(&condnonvalid, &mut_var);
            share = (int)rand();
            if (share == -1) share = 0;
            if (i == ITEMS - 1) share = -1;
            printf("Process %s: %i\n", (char *)a, share);
            valid = VALID;
            pthread_cond_signal(&condvalid);
            pthread_mutex_unlock(&mut_var);
      };
      printf("Process %s: end\n", (char *)a);
      return NULL;
}

int main(){
      pthread_t a,b;
      pthread_mutex_init(&mut_var, NULL);
      pthread_cond_init(&condvalid, NULL);
      pthread_create(&a, NULL, producent, "producent");
      pthread_create(&b, NULL, konzument, "konzument");
      pthread_join(a, NULL);
      pthread_join(b, NULL);
      pthread_cond_destroy(&condvalid);
      pthread_cond_destroy(&condnonvalid);
      pthread_mutex_destroy(&mut_var);
      return 0;
}
```

Výpis č. 3: Použití mutexu v problému producent – konzument

je sdílen mezi procesy (hodnota != 0). V Linuxu jsou podporovány pouze lokální semafory.

```
int sem_wait(sem_t * sem);
```

- slouží pro vstup do kritické sekce. Pokud je sekce obsazena (semafor == 0), pak se čeká až se sekce uvolní.

```
int sem_trywait(sem_t * sem);
```

- slouží pro vstup do kritické sekce. Je-li sekce obsazena, funkce se vrátí s chybou EAGAIN.

```
int sem_post(sem_t * sem);
```

- slouží k ukončení kritické sekce.

```
int sem_getvalue(sem_t * sem, int * sval);
```

- vrátí hodnotu semaforu.

```
int sem_destroy(sem_t * sem);
```

- uvolní všechny zdroje spojené se semaforem.

Toto rozhraní pro semafory definuje norma POSIX 1003.1b a POSIX 1003.1i.

Ukončení vlákna jiným vláknem

```
int pthread_cancel(pthread_t thread);
```

- vyvolá požadavek na zrušení vlákna.

```
int pthread_setcancelstate(int state,
      int *oldstate);
```

- nastaví chování vlákna, které tuto funkci vyvolalo, na požadavek jeho zrušení. Možné jsou dva stavy: PTHREAD_CANCEL_ENABLE a PTHREAD_CANCEL_DISABLE.

```
int pthread_setcanceltype(int type, int *oldtype);
```

- nastaví, kdy je možno vlákno zrušit. Možné jsou dvě nastavení: PTHREAD_CANCEL_ASYNCHRONOUS — vlákno bude zrušeno skoro okamžitě po přijetí požadavku nebo PTHREAD_CANCEL_DEFERRED — vlákno se zruší až v okamžiku, kdy dojde do bodu, kde je možno vlákno zrušit. Jako body jsou v POSIXu definovány tyto funkce: pthread_join(3), pthread_cond_wait(3), pthread_cond_timedwait(3), pthread_testcancel(3), sem_wait(3), sigwait(3).

```
void pthread_testcancel(void);
```

- tato funkce pouze testuje, zda byl přijat požadavek na zrušení vlákna. Pokud přijat byl, vlákno je zrušeno, v opačném případě se funkce normálně vrátí. Funkce se používá v místech, kde jsou dlouhé kusy kódu bez bodů vhodných pro zrušení.

Pokud je vlákno v bodu vhodném pro zrušení (viz pthread_setcanceltype(3)) a přijalo požadavek na zrušení, bude zrušeno. Totéž se stane, pokud přijalo požadavek na zrušení a až následně vejde do bodu vhodného pro zrušení (pouze při nastaveném PTHREAD_CANCEL_DEFERRED).

Pokud se ukončí hlavní vlákno, aniž by počkalo na vlákna jím vytvořená, jsou tato vlákna ukončena také.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#define ITEMS 100

sem_t semfull;
sem_t semempty;
int share;

void * konzument(void *a){
  printf("Process %s: start\n", (char *)a);
  while(1){
    sem_wait(&semfull);
    printf("Process %s: %i\n", (char *)a, share);
    if (share == -1){
      sem_post(&semempty);
      break;
    };
    sem_post(&semempty);
  };
  printf("Process %s: end\n", (char *)a);
  return NULL;
}

void * producent(void *a){
  int i;
  printf("Process %s: start\n", (char *)a);
  for (i = 0; i<ITEMS; i++){
    sem_wait(&semempty);
    share = (int)rand();
    if (share == -1) share = 0;
    if (i == ITEMS - 1) share = -1;
    printf("Process %s: %i\n", (char *)a, share);
    sem_post(&semfull);
  };
  printf("Process %s: end\n", (char *)a);
  return NULL;
}

int main(){
      pthread_t a,b;

      sem_init(&semfull, 0, 0);
      sem_init(&semempty, 0, 1);
      pthread_create(&a, NULL, producent,
          "producent");
      pthread_create(&b, NULL, konzument,
          "konzument");
      pthread_join(a, NULL);
      pthread_join(b, NULL);
      sem_destroy(&semfull);
      sem_destroy(&semempty);
      return 0;
}
```

Výpis č. 4: Použití semaforů u problému producent – konzument

Jak použít funkce můžete vidět na příkladu Ukončení vlákna.

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <semaphore.h>

sem_t sem;

void * process(void *a){
   printf("Proces entered\n");
   sem_wait(&sem);
   printf("Proces exiting\n");
   return NULL;
}

int main(){
   pthread_t thid;
   void * thretval;

   sem_init(&sem, 0, 0);

   pthread_create(&thid, NULL, process, NULL);

   sleep(5); /* Vlákno se zatím rozběhne */
   if (pthread_cancel(thid) != 0)
     printf("Cancel error\n");

   pthread_join(thid[i], &thretval);
   if (thretval != PTHREAD_CANCELED)
     printf("Thread not be canceled.\n", i);

   return 0;
}
```

Výpis č. 5: Ukončení vlákna

Další užitečné funkce

```
pthread_t pthread_self(void);
```

- vrací identifikátor vlákna, které tuto funkci vyvolalo.

```
int pthread_equal(pthread_t thread1,
      pthread_t thread2);
```

- porovná, zda se identifikátory vláken rovnají.

```
int pthread_detach(pthread_t th);
```

- odpojí vlákno. Všechny paměťové prostředky, které vlákno používá, budou po ukončení vlákna okamžitě uvolněny. S odpojeným vláknem se nelze synchronizovat a vyzvednout jeho návratový kód funkcí `pthread_join(3)`.

```
int pthread_attr_init(pthread_attr_t *attr);
```

- inicializuje objekt atributů na implicitní hodnoty. Tento objekt se dá použít pro vytvoření více vláken.

```
int pthread_attr_destroy(pthread_attr_t *attr);
```

- uvolní všechny prostředky potřebné pro objekt atributů.

Problémy, do kterých se můžete dostat

- knihovna pro vlákna používá signály `SIGUSR1` a `SIGUSR2`, proto je program používat nemůže.
- pokud budete vlákna používat v X aplikacích, je třeba mít `Xlib` kompilovánu s `-D_REENTRANT` a podporou vláken (knihovna musí být napsána reentrantně — více vláken může vykonávat tutéž funkci ve stejnou chvíli, bez vzájemného ovlivnění — funkce nepoužívají globální proměnné). Totéž platí o jakékoliv knihovně, kterou budete v programu používat. Pokud knihovna takto reentrantní není, je možno ji používat, ale pouze z hlavního vlákna (kódu procesu). Toto souvisí s proměnnou `errno`. Každé vlákno má totiž vlastní, pouze hlavní vlákno používá globální `errno`.
- používání vláken v C++ s `libg++` asi nebude fungovat. Pro používání vláken v C++ je doporučen překladač `egcs` a knihovna `libstdc++`.
- pokud program vytvoří například 2 vlákna, nedivte se, že vidíte 4 stejné procesy. Jeden je hlavní proces, pak vidíte 2 vlákna a poslední je vlákno starající se o správný chod vláken. Toto vlákno je vytvořeno knihovnou `pthread`.

Odkazy

Na adrese (1) lze najít často kladené otázky newsové skupiny *comp.programming.threads*.

Bližší informace o linuxových vláknech naleznete na adrese (2). Lze zde také najít tutorial.

Na adrese (3) najdete X/Open Group Single Unix specification, kde by se měl také dát najít bližší popis aplikačního rozhraní pro vlákna.

Na adrese (4) najdete projekt, který také usnadňuje používání vláken v C++. ■

1 FAQ comp.programming.threads
http://www.serpentine.com/~bos/threads-faq/
2 Linux Threads
http://pauillac.inria.fr/~xleroy/linuxthreads
3 Single Unix specification
http://www.rdg.opengroup.org/onlinepubs/7908799/index.html
4 Vlákna v C++
http://www.cs.wustl.edu/~schmidt/ACE.html

autoconf, automake a libtool aneb Makefile snadno a rychle

Henryk Paluch, 9. září 1998

Tyto tři pomůcky slouží k rychlé tvorbě souborů `Makefile`, snadné a na platformě nezávislé kompilaci knihoven, včetně dynamických, a také poskytují rozsáhlé možnosti při testování různých specifik systému, na kterém bude program kompilován.

- `libtool` unifikuje tvorbu dynamických a statických knihoven. Volá se místo překladače C a podle typu plat-

```
# Makefile.am - ukázkový program k libtool
AUTOMAKE_OPTIONS = foreign
# druh distribuce
# foreign - nejmírnější pravidla
# gnu     - musí být přítomny soubory README NEWS COPYNG aj.

# další soubory, které jsou součástí distribuce
EXTRA_DIST = aclocal.m4

# vytvořit knihovnu libhello pomoci libtool
# *.la je jméno knihovny, která byla/bude vytvořena pomocí libtool
lib_LTLIBRARIES = libhello.la
# zdrojové soubory knihovny libhello
libhello_la_SOURCES = hello.c foo.c
# číslování verzí je specifické pro libtool, viz. manuál
libhello_la_LDFLAGS = -version-info 3:12:1

# hlavičkové soubory, které se mají nainstalovat do
# ${prefix}/include, jinak je uved'te jen v *_SOURCES
include_HEADERS = foo.h

# spustitelné programy
bin_PROGRAMS = hell hell.static

# zdrojové soubory programu hell
hell_SOURCES = main.c
# knihovny
hell_LDADD = libhello.la

# totéž pro hell.static
hell_static_SOURCES = main.c
# pokud byste chtěli linkovat knihovnu, která nebyla vytvořena
# s libtool, tak použijte klasické libxxx.a místo libxxx.la
hell_static_LDADD = libhello.la
hell_static_LDFLAGS = -static
```

Výpis č. 6: Šablona Makefile `Makefile.am`

formy zajistí doplnění příkazové řádky potřebnými parametry pro kompilaci a instalaci staticky nebo dynamicky linkované knihovny.

- `automake` vytváří standardizované `Makefile.in` podle šablony `Makefile.am`. Od verze 1.2 umí také spolupracovat s utilitou `libtool`. Cílové soubory `Makefile` generované programem `autoconf` pak obsahují standardní metody pro kompilaci, instalaci a distribuci programu.

- `autoconf` generuje z šablony `configure.in` konfigurační skript `configure`. `autoconf` nám poskytuje bohaté možnosti k otestování systému. Můžeme snadno ověřit přítomnost různých programů, hlavičkových souborů a hlavně knihoven a funkcí. Máme k dispozici např. makra, která zjistí přítomnost X Window System a dodají všechny potřebné parametry, včetně cest k hlavičkovým souborům a knihovnám.

  Skript `configure` se spouští před samotnou kompilaci našeho programu. Jeho úkolem je otestovat systém, vytvořit `Makefile` z `Makefile.in` a provést v něm substituci proměnných. Další činnost tohoto skriptu závisí pochopitelně na šabloně `configure.in` — např. konfigurace `libtool` apod.

Instalace

Měli byste nainstalovat tyto RPM balíky (součást distribuce Red Hat Linux 5.1):

```
# rpm -Uvh autoconf-2.12-3.noarch.rpm \
           automake-1.3-2.noarch.rpm  \
           libtool-1.0h-2.noarch.rpm
```

Tyto programy na sobě do určité míry závisejí. Pokud si je budete kompilovat sami, ujistěte se, že všechny mají stejný instalační `--prefix`!

Příklad

Uvedený příklad pochází z ukázkového programu `hell-o` ke skriptu `libtool`.

Obsahuje tyto soubory:

- `hello.c` a `foo.c` jsou součástí knihovny `libhello`
- `main.c` je testovací program, který využívá výše zmíněnou knihovnu

Generování skriptů

Poprvé zadáme tuto sekvenci:

```
dnl AC_INIT - vždy 1. příkaz
dnl parametr - jméno souboru, který je souč. dist
dnl   pro kontrolu
AC_INIT(hello.c)
dnl jméno balíku a číslo verze
AM_INIT_AUTOMAKE(hell,1.0)

dnl otestuje překladač C
AC_PROG_CC
dnl nakonfiguruje libtool
AM_PROG_LIBTOOL

dnl otestuje přítomnost funkce cos() v knihovně
dnl libm.a (resp. libm.so.*)
AC_CHECK_LIB(m, cos)
dnl v případě úspěchu definuje -DHAVE_LIBM

dnl Vygeneruje Makefile z Makefile.in
dnl a provede substituci proměnných
AC_OUTPUT(Makefile)
```

Výpis č. 7: Šablona autoconfu `configure.in`

- `$ libtoolize` — vytvoří `libtool` a další pomocné soubory
- `$ automake -a` — vygeneruje `Makefile.in`. Volba `-a` navíc zajistí přidání chybějících souborů
- `$ aclocal` — přidá makra `AM_INIT_AUTOMAKE` a `AM_PROG_LIBTOOL` do `aclocal.m4`.
- `$ autoconf` — vygeneruje `configure` z `configure.in`.

Pokud jen modifikujeme např. `Makefile.am`, pak stačí zadat:

```
$ automake; aclocal; autoconf;
```

To už však není nezbytně nutné: `Makefile` obsahuje všechny potřebné závislosti a všechny konfigurační soubory se znovu vygenerují automaticky.

Kompilace

- `$ ./configure` — vytvoří `Makefile`. Pokud se objeví chybové hlášení `AM_INIT_AUTOMAKE: File not found`, znamená to, že `aclocal.m4` toto makro neobsahuje. Ujistěte se, že všechny tři balíky byly instalovány se stejným prefixem.
- `$ make` — zkompiluje program `hell` a knihovnu `libhello`
- `$ ./hell` — spustí dynamicky linkovaný hell
- `$ ./hell.static` — spustí staticky linkovaný hell
- `$ make install` — nainstaluje zkompilované programy a knihovny
- `$ make dist` — vygeneruje balík vhodný pro distribuci zdrojových textů. Ke kompilaci tohoto balíku nejsou programy `autoconf`, `automake`, a `libtool` vůbec potřebné — ty slouží pouze k vytvoření konfiguračních skriptů.
- `$ make dist-all` — vygeneruje kompletní archív
- `$ make clean` — vyčistí projekt

Závěr

Tento příspěvek jen nastínil možnosti, které tyto programy nabízejí. Pokud se rozhodnete je používat, zcela jistě vám budou k užitku referenční `info` manuály (1), (2) a (3). Buďte připraveni na různé problémy a úskalí, která vás mohou potkat. Zvláště `automake` je někdy poněkud tvrdohlavý a nezbyde vám, než se přizpůsobit jeho stylu. Jako studijní materiál vřele doporučuji skripty z jiných programů (GIMP, gtk, SANE). ■

1 David MacKenzie
Autoconf --- Creating Automatic Configuration Scripts
2 David MacKenzie, Tom Tromey
GNU Automake
3 Gordon Matzigkeit
GNU Libtool

Tvorba RPM balíků — pokračujeme

Jan Kasprzak, 25. srpna 1998

V minulých číslech Linuxových novin jsme si ukázali základní sekce `spec`-souboru. Nyní budeme pokračovat popisem skriptů, které ve `spec`-souborech mohou být.

Instalační skripty

Jak již bylo několikrát řečeno, RPM balík může obsahovat skripty, které se spustí před instalací balíku, po instalaci, případně před nebo po odinstalování balíku. Všechny tyto skripty dostanou od RPM proměnnou prostředí `RPM_INSTALL_PREFIX` udávající, do kterého adresáře se balík instaluje. Tato proměnná se použije pouze u balíků, které nemusí být instalovány v pevné adresářové struktuře (mohou bez problémů být například v `/usr/local` nebo v `/opt`. Příslušné skripty jsou ve `spec`-souboru uvozeny jedním z následujících slov podle toho, o který typ jde:

```
%pre
%preun
%post
%postun
```

Samozřejmě ne všechny čtyři skripty je nutno uvést. Tyto skripty jsou jen jako výpomoc pro případy, kdy nestačí pouze nainstalovat soubory daného balíku, ale je ještě potřeba udělat něco dalšího (například přidat resp. odebrat záznam z `/etc/shells`, jde-li o balík shellu).

Pokud upgradujeme daný balík, jsou spuštěny nejprve pre- a post-instalační skripty z nové verze, a potom teprve pre- a post-uninstall skripty ze starší verze, jak lze poznat například napsáním malého RPM balíku s testovacími výpisy v těchto skriptech:

```
# rpm -U a-2-2.noarch.rpm
Here is a pre v2 script 2.
Here is a post v2 script 2.
Here is a preun v1 script 1.
Here is a postun v1 script 1.
```

Jak rozpoznat instalaci od upgrade?

Často ve zmiňovaných skriptech chceme vykonat nějakou akci pouze v případě, kdy jde o první instalaci daného balíku (tedy nikoliv při upgradu). Příkladem může být přidání (pseudo)uživatele, kterého náš balík potřebuje ke své činnosti. V tomto konkrétním případě je sice možné podívat se do `/etc/passwd`, jestli tam už tento uživatel není, ale jsou i případy, kdy tato detekce není možná nebo je příliš náročná. Pro příklad nemusíme chodit daleko — stačí vzít post-uninstall skript, ve kterém chceme zrušit uživatele z `/etc/passwd`. Jak zjistit, jestli jde o skutečné rušení balíku, nebo jestli se jen post-uninstall skript volá proto, že je systém právě upgradován na novější verzi balíku?

Systém RPM pro tento případ nabízí skriptům číselný argument (dostupný ze shellu jako `$1`), který říká, kolik instancí daného balíku bude v systému po dokončení akce nad aktuálním balíkem. V případě instalace tedy `%post` a `%pre` skripty dostanou parametr 1, zatímco v případě upgrade je parametr 2. Obdobně skripty `%postun` a `%preun` dostanou v případě upgrade parametr 1, zatímco v případě úplného rušení balíku parametr 0. Náš testovací balík z předchozího odstavce vypisuje parametr skriptu jako poslední slovo. Pokud tento balík odinstalujeme, dostáváme následující:

```
# rpm -e a
Here is a preun v2 script 0.
Here is a postun v2 script 0.
```

Verifikační skript

Tento skript, uvozený slovem `%verifyscript` je spouštěn v okamžiku kontroly daného balíku (`rpm -V`). Je zde možné například kontrolovat, jestli je správně konfigurace balíku, jestli je (například) náš shell uveden v `/etc/shells`, jsou-li vytvořeni příslušní uživatelé a podobně.

Verifikační skript, stejně jako čtyři předchozí skripty, je spouštěn programem `/bin/sh`. Pokud chceme mít skript v jiném programovacím jazyce, je možno použít přepínač `-p interpret`, například:

```
%pre -n /usr/bin/perl
print "Just another perl hacker.\n";
```

Spouště

Pod slovem v nadpisu se skrývá překlad anglického *triggers*. Z databází známe spouště jako procedury, které se spustí v okamžiku modifikace některé určité databázové položky. V RPM je tato vlastnost horkou novinkou (a není ani popsána v Maximum RPM). Podrobné informace se lze dočíst na adrese (1).

Co jsou spouště a k čemu je lze použít? Jde o skript, který je součástí nějakého balíku, a který se spouští v případě, že systém RPM nějakým způsobem manipuluje (instaluje, upgraduje, ruší) s jiným balíkem. Ve zmiňované dokumentaci je příklad, kdy nějaký hypotetický poštovní klient má svůj symbolický link `/etc/mymail/mymailer`, který ukazuje na příslušný MTA (mail transport agent), nainstalovaný v systému. A jsou zde uvedeny dvě spouště — jedna pro balík `sendmail` (2) a druhá pro `vmailer` (3). A tyto skripty mají právě na starost vytváření onoho linku.

Tento příklad je dosti umělý, protože poštovní klienti umějí buďto SMTP, nebo umějí nový mail dát na vstup programu `/usr/sbin/sendmail`. A MTA zase obsahuje program, link nebo skript `/usr/sbin/sendmail`, který slouží pro odeslání pošty, a MTA umí i SMTP. Já jsem s úspěchem použil spoušť v balíku `groff-latin2`, kdy je potřeba modifikovat konfigurační soubor `groffu`. Takže mám skript, který toto udělá vždy, když je `groff` instalován nebo upgradován. Mechanismus spouští se vůbec dobře hodí pro balíky, které modifikují soubory, patřící jiným balíkům.

Jak se použije spoušť? Jde o shellovský (nebo i jiný) skript, uvozený jedním z následujících slov:

```
%trigger
%triggerin
%triggerun
%triggerpostun
```

Za tímto slovem jsou uvedeny přepínače (například `-p interpret`), následují dva mínusy (`--`) a pak specifikace tzv. „cílového“ balíku, tedy balíku, pro který se má daná spoušť spustit. Specifikace má stejnou syntaxi jako tag `Requires:` v hlavičce `spec`-souboru. Můj balík `groff-latin2` například má tuto spoušť:

```
%trigger -- groff
```

Složitější spoušť by mohla být uvozena takto:

```
%trigger -p /usr/bin/perl -- balík2 >= 2.1.117
```

Kdy se spoušť spustí?

- %trigger
  - zdrojový balík se instaluje nebo upgraduje a cílový je již nainstalován. Spouští se po `%post` skriptu (nového) zdrojového balíku.
  - cílový balík se instaluje nebo upgraduje a zdrojový je již nainstalován. Spouští se po `%post` skriptu (nového) cílového balíku.
- %triggerin
  - je ekvivalentní s `%trigger`
- %triggerun
  - zdrojový balík se právě ruší nebo upgraduje a cílový je nainstalován. Spouští se před `%preun` skriptem (původního) zdrojového balíku.
  - cílový balík se ruší nebo upgraduje a zdrojový je nainstalován. Spouští se před `%preun` skriptem (původního) cílového balíku.
- %triggerpostun
  - cílový balík se ruší nebo upgraduje a zdrojový je nainstalován. Spouští se po `%postun` skriptu (původního) cílového balíku.

V příští části si povíme o tom, jak vytvářet seznam souborů obsažených v RPM balíku a také o problematice maker. ■

1 Triggery v RPM
http://www.rpm.org/support/RPM-Changes-6.html

2 Sendmail
http://www.sendmail.org
3 Vmailer
http://www.porcupine.org/vmailer

Mají tučňáci žluté nohy?

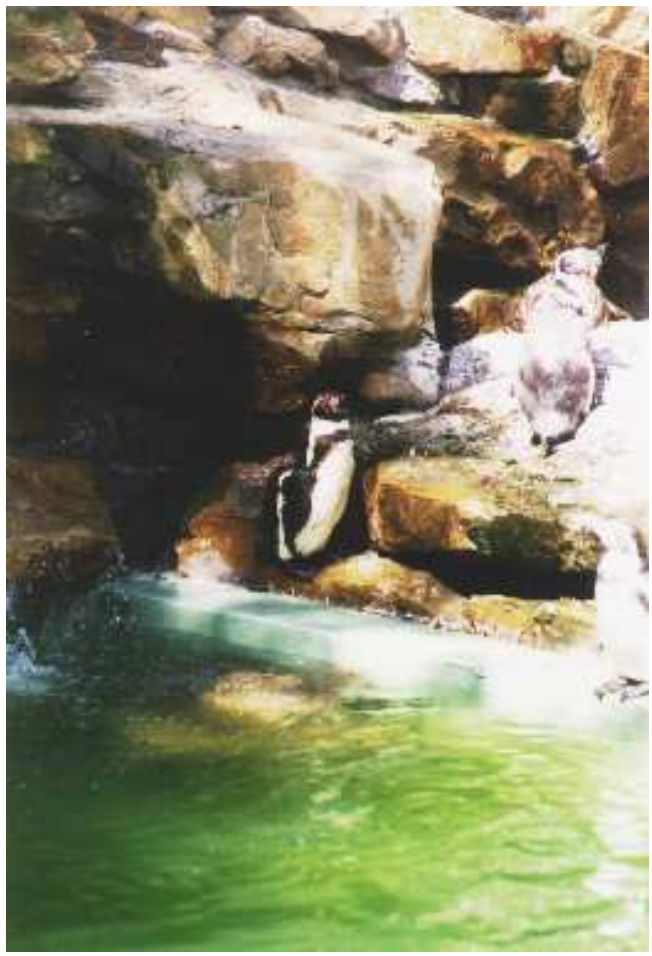
Pavlína Vavrečková, 2. září 1998

V pondělí 3. 8. 1998 proběhla další akce CZLUGu s názvem „Mají tučňáci žluté nohy?“. Na programu byla návštěva ZOO Lešná, kde letos v květnu otevřeli novou expozici tučňáků Humboldtových. Cílem — jak sám název akce napovídá — bylo podrobněji se seznámit se životem těchto tvorů a hlavně zjistit, jak je to vlastně s barvou tučňáčích nohou (problém, který nedá spát Pavlu Janíkovi).

Jednalo se o první akci v historii CZLUGu, kde byla převaha něžného pohlaví, a to v poměru 3:2. Všichni účastníci byli vybaveni kultovními linuxovými tričky. Výjimku tvořila jedna slečna, která měla tričko s obrovskou žábou a nápisem „rrrriiiibít“ (to prý dělají anglické žáby).

Hned po vstupu do ZOO jsme se snažili dostat co nejrychleji k tučňákům. Jen zběžně jsme si prohlédli lachtany a plni očekávání pokračovali ve směru šipek s obrázky tučňáků. „…již zdálky bylo slyšet mumlavé zvuky, které postupně sílily, a za chvíli už jsme spatřili tučňáky skandující „Linux“ a vítající Yenyu.“ (Převzato od Rohlíka). No dobře, trochu si vymýšlím, ve skutečnosti to bylo jinak: tučňáci rozpačitě přešlapovali a vydávali zvuky, které podle přítomných linuxových odborníků vzdáleně připomínaly slovo „Linux“.

Po našem příchodu se všichni tučňáci zdržovali v zadní části výběhu, kde je vybudován systém jeskyněk s rozprašovači. Dále mají tučňáci k dispozici dva bazény spojené potůčkem. Součástí horního bazénu je dokonce mini-vodopád. Dolní bazén se nachází nad úrovní terénu a má průhledné stěny, takže za optimálních podmínek je v něm možno pozorovat chování tučňáků pod vodou. My jsme bohužel takové štěstí neměli — na dolní bazén totiž svítilo sluníčko, takže tučňáky ani nenapadlo se do něj přemisťovat. Když už jsme stáli u výběhu dostatečně dlouho, začalo to být tučňákům podezřelé. Uspořádali proto něco jako malý sněm, kterému velel starší pelichající tučňák.



Mezitím co se tučňáci radili, dočetli jsme se, že byli darováni curyšskou ZOO, a že jsou velmi citliví na znečištění ovzduší.

Po chvíli se všichni tučňáci začali plácat křídlem do čela — správně pochopili, že se od nich ještě něco očekává. Takže nám začali předvádět své plavecké a potápěčské umění. A my jsme si mohli poznamenat další poznatek ze života tučňáků: plovoucí tučňák vypadá jako kačena a umí se děsně hluboko potápět.

Představení, které pro nás tučňáci připravili, bylo završeno krmením. Krmení tučňáků probíhá tak, že přijde člověk s miskou ryb a snaží se je rovnoměrně rozdělit mezi všechny zúčastněné tučňáky. Tučňák zachytivší rybu s ní vždy skočí do bazénu. Možná si chce namluvit, že ji skutečně ulovil ve vodě, možná se nehodlá dělit s ostatními. Těž-

ko říct. Tento rys tučňáčího života si ještě zaslouží hlubší pozorování. Jeden z tučňáků se projevil jako zvlášť inteligentní, když se snažil ostatní přelstít tím, že nejprve skočil do vody bez rybičky a poté vylezl na břeh přímo u zřízence s potravou. Iluze inteligence by byla dokonalá, kdyby se tak výše zmíněný tučňák nechoval ještě půl hodiny poté, co člověk s krmením odešel.

Teď, jen tak pro zajímavost — srovnání pražských tučňáků (ty jsme navštívili loni) se zlínskými: tučňáci v Praze vylézají z vody tak, že začnou děsně rychle plavat proti břehu, takže to vypadá, jakoby z vody vyběhli. Do vody se dostanou tak, že tam prostě rovnou skočí. Tučňáci v Lešné si při vylézání z vody pomáhají křídlem. Do vody se vzájemně shazují. No jo, jiný kraj, jiný mrav.

A k jakému závěru jsme dospěli? Většina účastníků se shodla na tom, že mezi nám známými tučňáky převládají tučňáci se žlutýma nohama. Tuto hypotézu potvrdil i tučňák, kterého jsme minulý týden objevili v brněnském Tescu: je sice plyšový, ale je velký a má žluté nohy!!! (Yenya mi ho nechce koupit, protože se bojí, že by ho vytlačoval z betle :-() ■

Jaký byl Linux InstallFest?

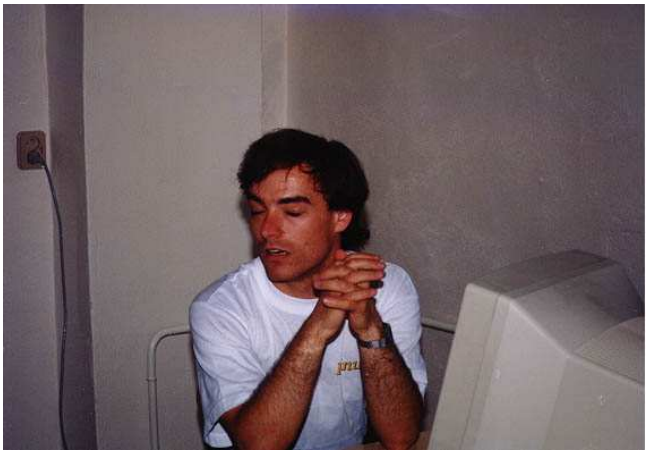
Pavel Janík ml., 31. srpna 1998

Tak jaký byl? Únavný a zajímavý, zničující a poučný, dlouhý i krátký. Protože se jeho organizátoři z něj ještě nevzpamatovali, přinášíme vám pouze fotografie, které Linuxovým novinám laskavě poskytl pan Mirek Prokop. Ostatně pokud si chcete přečíst více o průběhu celé akce, stačí zabloudit na Svět Namodro (1) a chvíli hledat.

V úvodu celé akce jsme mohli vyslechnout přednášku Jana „Yenyi“ Kasprzaka o hlasových modemech.

A poté se to vše zvrhlo — instalovalo se a instalovalo a instalovalo.

Představitel projektu Debian CZ/SK (2) Milan Zamazal neskrývaně pózuje s právě vypáleným CDčkem s nejnovější počeštěnou distribucí operačního systému Red Hat Linux 5.1, kterou doma ihned rozebere a vše jistě doplní do projektu Debian CZ/SK.



Na konci akce toho ale někteří účastníci měli opravdu dost — Yenya se např. modlil a v duchu si opakoval jedinou větu: „CENZUROVÁNO — kas@informatics.muni.cz“.

Více vám toho ale opravdu neprozradím, měli jste tam být s námi :-). Takže snad příště. ■

1 Svět Namodro
http://svet.namodro.cz
2 Debian CZ/SK
http://www.debian.cz

Z druhého břehu

Miroslava Krátká, 2. září 1998

Byla to pěkná letní sobota. Slunce svítilo a několik nadšenců se po poledni blížilo k fakultě informatiky MU. Někteří jen nesměle nakukovali do otevřené učebny, kde ještě probíhaly poslední úpravy, ti méně stydliví se zeptali a mohli si pak vybrat to nejlepší místo pro sebe, případně pro svého hranatého miláčka (prosím pány o prominutí mého poněkud neodborného termínu).

Krátce po jedné si návštěvníci vyslechli zajímavou přednášku (tedy alespoň podle výrazu jejich tváří se zdálo, že je zajímá). A potom se instalovalo. Tu byla potřeba disketka, tu síťová karta, jinde zase chyběl šroubováček, někomu stačila jen malá rada. Vše bylo k zapůjčení pouze za podpis, rady poskytované Pavlem Janíkem a Yenyou i bez podpisu.

Otevřené knihkupectví skýtalo možnost zakoupení odborných publikací, CD, ale také triček s tučňáky. Ta měla velký úspěch. Někdo nám dokonce na chvilku poodhalil svoji hruď (nebo se k nám naopak otočil zády), jen aby měl co nejdříve toto roztomilé žlutonohé zvířátko blíže svému srdci. Jiný byl na tričko dokonce tak hrdý, že si ho oblékl obráceně (asi proto, aby měl většího tučňáka vpředu).

Závěrem bych si dovolila malý osobní poznatek. Na počátku mého středoškolského studia jsem si myslela, že do počítačové učebny se „vejde“ jen tolik posluchačů, pro kolik jsou zde klávesnice. Studia na gymnáziu a vysoké škole tento můj názor velmi ovlivnila. Důležitý je počet židlí, nezávisle na tom, kolik párů očí potom sleduje (nebo se o to snaží) jeden monitor. Nyní je mi to již jasnější. Počítačová učebna je totiž nafukovací. Pokud nestačí židle, odněkud se donesou; stůl také není potřeba, vždyť notebook klidně „posedí“ na klíně…

Toť mé zážitky z Linux InstallFestu. ■

Zasmáli jsme se!

Pavel Janík ml., 31. srpna 1998

Samozřejmě i o prázdninách je lidský humor (u nás se tomu říká sranda) součástí každodenní přítomnosti na Síti a proto bych se s vámi — čtenáři Linuxových novin — rád podělil o úsměvné okamžiky strávené nad mojí poštovní přihrádkou.

Začněme v domácích luzích a hájích …Asi nejzajímavější signaturou v konferenci *linux@muni.cz* je:

> Je libo Win98? — Ne, díky, počkám si na Win00.
>
> *signatura Ivo Žáčka*

Ale samozřejmě i v zahraničí používají signatury:

> LinuxScan report : Win95 virus found on boot sector of primary partititon.
> Severity : medium, it could halt your computer at random time, corrupt your datas and hard disk or try to be reinstalled every two days.
> Disinfectant exists, apply (Yes/Yes/Yes) ?
>
> *Pascal A. Dupuis*
> *v linux-kernel@vger.rutgers.edu*

Pokračujme pouze takovou malou zajímavostí, kterou můžete také potkat. Richard B. Johnson, aktivní to čtenář listu *linux-kernel@vger.rutgers.edu*, si povšiml, že se mu poněkud záhadně zmenšuje dostupný diskový prostor na jeho pracovní stanici. Po dlouhém pátrání zjistil následující věc:

```
# ls -la /var/spool/mail
total 71010
drwxrwxrwx  2 ...       1024 Aug  5 09:22 .
drwxr-xr-x 19 ...       1024 Jul 22 13:29 ..
-rw-------  1 ... 1972425472 Aug  5 09:23 ftp
-rw-------  1 ...          0 Jul 13 12:51 johnson
-rw-------  1 ...          0 Aug  5 09:20 root
```

Inu spammeři jsou nevypočitatelní…

Já jsem měl v poslední době také podobnou příhodu — Dušan Dobeš (autor šachového programu `phalanx`) si mi stěžoval, že na našem školním serveru již nejdou spustit žádné další aplikace (`Unable to load interpreter` jste již jistě také někdy viděli). Po drobném zkoumání jsem zjistil, že program Netscape Communicator se ne dosta-

tečně dobře vyrovnal s mým požadavkem na stažení oficiálního image Debianu 2.0 (`binary-i386.raw`) o velikosti něco kolem 600 MB a těsně před tím, než byl korektně ukončen, spotřebovával 87% procesorového času a 613 MB paměti.

Na adresu konference *lesstif@hungry.com* věnovanou Lesstifu (Pro nezasvěcené: jedná se o free knihovnu přípravků, která si klade za cíl být kompatibilní s komerční knihovnou Motif jak na úrovni zdrojových textů, tak na úrovni vzhledu výsledných aplikací.) poslal nějaký téměř šílený Američan vystupující pod pseudonymem `Indl` následující příspěvek.

> why another similar widget set? why not something new, something you have not copied from an american? you have a pompus attitude towards americans, why copy them? why? why? why? I am an american, (your web site states you are some sort of hungarian effort,) I am tired of seeing the same widgets day in and day out, programming with the same ***** widget sets, etc... What is the point? You are quite strange to be only copying some one else's work, ya know.
>
> ---
>
> Proč další takový soubor přípravků? Proč něco dalšího, co jste okopírovali od Američana? Když zaujímáte k Američanům takový postoj, proč je kopírujete? Proč? Proč? Proč? Jsem Američan (váš webový server na vás prozrazuje, že vaše snahy mají kořeny v Maďarsku) a jsem unaven z toho, že den co den vidím ty samé přípravky, používám ty samé ***** přípravky, atd. Co to je za nápad? Stačíte jen na to, abyste kopírovali něčí dílo, co.

Blahoslaveni chudí duchem. Jeden z nejaktivnějších přispěvatelů Jon Christopher však odpověděl naprosto famózně. Jen mě přitom napadlo, jaké konce by vzal podobný útok vedený proti Linuxu na *linux@muni.cz*.

> You're absolutely right. We don't want to be doing this, but we're being held prisoner by the Hungarian mafia, and forced to write widget sets. Please help us regain our freedom by contributing code to our effort. And we don't *want* to be pompus to Americans, but the Mafia doesn't like Americans and forces us to say those awful things.
>
> ---
>
> Máte naprosto pravdu. My to dělat nechceme, ale jsme v zajetí maďarské mafie, která nás nutí psát knihovny přípravků. Prosíme, pomozte nám získat svobodu tím, že podpoříte naše dílo nějakým kódem. *Nechceme* být proti Američanům, ale mafie je nesnáší a nutí nás říkat takové ošklivé řeči.

Když jsem četl jakousi konferenci, do které přišel dopis se subjectem „Windows 98 Source code released“, nevěřil jsem (Windows 98 Source Code). Když jsem si ale porovnával zdrojový text s výsledkem, uvěřil jsem...

Pokud sledujete vývoj linuxového jádra a průběžně si kompilujete nové kernely pro své dvou či víceprocesorové stroje (pokud tento text někdo takový čte, rád bych jej požádal o zaměstnání `:-)`) jistě jste si povšimli, že vám jádro 2.1.112 nechodilo zrovna tak, jak byste si představovali. Inu, co vše se muže stát, když používáte metodu *cut-and-paste* pro patchování kernelu vysvětlil Linus Torvalds takto:

> That should get 2.1.112 working again (and should teach me not to do final touch-ups on code after I've tested it but before I release it even if those touch-ups look obvious, but it's hard to teach an old dog new tricks...)
>
> ---
>
> To by mělo verzi 2.1.112 opět zprovoznit (a mělo by mne to poučit, že se už nemám dotýkat kódu, který jsem už otestoval, ale ještě nezveřejnil, i přesto, že tyto „doteky“ mohou být zřejmé, ale učit starého psa novým kouskům je obtížné...)
>
> *Linus Torvalds ve vysvětlení chybičky v 2.1.112*

A když už jsme u Linuse, tak vám doporučím přečíst si jeden článek (1), kde je Linus citován:

> I knew I was the best programmer in the world. Every 21-year-old programmer knows that. "How hard can it be, it's just an operating system?"
>
> ---
>
> Věděl jsem, že jsem nejlepším programátorem na světě.
> Každý programátor ve věku 21 let to ví.
> „Jak to může být těžké, když je to jen operační systém?“
>
> *Linus Torvalds*

Ale to byla pouze taková malá odbočka co se může a nemůže stát. Příště již snad budete také na Síti a přispějete svými dopisy i do této rubriky Linuxových novin. ■

1 ComputerWorld: Meet Linus Torvalds
`http://www.computerworld.com/home/features.nsf/all/980817linus`

A co příště?

Pavel Janík ml., 13. září 1998

Jak jistě víte, za několik málo týdnů je zde výstava INVEX, která se koná v Brně. V příštím čísle se vám tedy pokusíme představit alespoň několik produktů, které na INVEXu nabízeli vystavovatelé pro Linux (tajně doufám, že se nám podaří získat demo verze všech).

Chtěli bychom vám také ukázat, jak je možné i na Linuxu vytvářet kvalitní dokumenty, které mohou číst nejenom uživatelé Linuxu, ale také např. Windows či Macintoshů. Ano, ukážeme vám program pdfTEX, jehož autorem je Han The Thanh.

Samozřejmě budou také pokračovat všechny seriály a doufejme, že toho bude mnohem více. ■

```
/* The Win98 source code has been released recently; Per MicroSoft
   CopyRights, you can only view the code, but should not change it! */
        /*   TOP SECRET Microsoft(c)  Code
             Project: Chicago(tm)
             Projected release-date: Summer 1994   */
         #include "win31.h"
         #include "win95.h"
         #include "evenmore.h"
         #include "oldstuff.h"
         #include "billrulz.h"
         #define INSTALL = HARD
  char make_prog_look_big[1600000];
         void main()
         {
             while(!CRASHED)
             {
                 display_copyright_message();
                 display_bill_rules_message();
                 do_nothing_loop();
                 if (first_time_installation)
                 {
                     make_50_megabyte_swapfile();
                     do_nothing_loop();
                     totally_screw_up_HPFS_file_system();
                     search_and_destroy_the_rest_of_OS/2();
                     hang_system();
                 }
                 write_something(anything);
                 display_copyright_message();
                 do_nothing_loop();
                 do_some_stuff();
                 if (still_not_crashed)
                 {
                     display_copyright_message();   do_nothing_loop();
                     basically_run_windows_3.1();   do_nothing_loop();
                     do_nothing_loop();
                 }
             }
     if (detect_cache())
                 disable_cache();
     if (fast_cpu())
             {
                 set_wait_states(lots);       set_mouse(speed, very_slow);
                 set_mouse(action, jumpy);    set_mouse(reaction, sometimes);
             }
             /* printf("Welcome to Windows 3.11"); */
             /* printf("Welcome to Windows 95"); */
    printf("Welcome to Windows 98");
     if (system_ok())
                  crash(to_dos_prompt);
             else
                 system_memory = open("a:\swp0001.swp", O_CREATE);

             while(something)
             {
                 sleep(5);  get_user_input();
                 sleep(5);  act_on_user_input();
                 sleep(5);
             }
             create_general_protection_fault();
         }
```

Výpis č. 8: Windows 98 Source Code

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 IBM Global Network na adrese (4), na serveru Gymnázia Vídeňská v Brně (5), na serveru časopisu Netáčik (6), který je připojen do slovenského SIXu, případně na serveru Mathew (7).

Linuxové noviny jsou k dispozici také ve formátu HTML na adrese (8). ■

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 Network
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 — Mathew
http://www.mathew.sk/noviny
8 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@pdas.cz
převod do HTML: Pavel Juran
mailto:xjuran@cs.felk.cvut.cz