Květen 1998

Linuxové noviny

Úvodem

Pavel Janík ml., 16. května 1998

Společnost Corel Computer (1) má na své úvodní stránce tučňáka. Databáze Interbase (2) je k dispozici pro Red Hat Linux 4.2, databáze Ingres (3) byla portována na Linux, Linus zveřejnil Linux verze 2.1.102. Tak tohle jsou nezajímavější novinky měsíce května.

V květnovém čísle Linuxových novin si můžete přečíst o výsledcích soutěže z minulého čísla Nejmenší vyhrává a také o tom, jak si Pavel Kaňkovský poradil se zadáním soutěže.

Milan Keršláger nás detailně seznámí s konfigurací IPX IPX v Linuxu a s programy Linware a Mars NetWare zdarma. Michal Polák měl k dispozici druhé vydání knihy Linux — Internet Server, a tak nám o tom něco napsal v článku LINUX — Internet server (druhé upravené vydání).

Samozřejmě nechybí již pravidelné rubriky Měsíc v comp.os.linux.announce, Co nového na sunsite.unc.edu?, Linux Journal a Zasmáli jsme se!. Navíc v květnovém čísle opět naleznete zajímavou soutěž v článku Co se stane, když. ■

1 Corel Computer
http://www.corelcomputer.com
2 Interbase pro Linux
http://www.interbase.com
3 Computer Associate – Ingres
http://www.ingres.com

Nejmenší vyhrává

Pavel Janík ml., 9. května 1998

V minulém čísle Linuxových novin byla vyhlášena soutěž o nejmenší spustitelný program ve formátu ELF. Soutěže se zúčastnilo sedm statečných, jejichž programy naleznete na stejných místech jako Linuxové noviny. Vítězem se stal Pavel Juran (mimochodem autor HTML podoby Linuxových novin), jehož implementace příkazu `true` má velikost celých 57 bytů. Pavel Kaňkovský zvolil jiný postup — implementoval program, který vrací počet zadaných argumentů na příkazové řádce. Pokud máte zájem o mini shell, jistě vás uspokojí program Milana Kopačky. Jeho `smallsh` má velikost 3176 bajtů. Velikosti soutěžních programů naleznete v tabulce.

Jméno	Velikost v bytech
Pavel Juran	57
Pavel Kaňkovský	60
Martin Mareš	72
Stanislav Meduna	124
Radovan Garabík	368
Henryk Paluch	794
Milan Kopačka	3176

Soutěže se zúčastnil i Martin Mareš. Kdo slyšel jeho přednášku na Linuxovém semináři na Cikháji, jistě si nenechá ujít zdrojový text jeho příspěvku — a dobře udělá…

Všechny příspěvky mohou být šířeny za podmínek licence GNU GPL. Třeba se najde někdo, kdo bude chtít vyrobit nějakou mini-distribuci.

Doufám, že se vám soutěž líbila a že soutěže, která je vypsána v tomto čísle Linuxových novin, se zúčastní více čtenářů. ■

Měsíc v comp.os.linux.announce

Pavel Janík ml., 1. května 1998

Společnost Dynamical Systems, Ltd. uveřejnila první verzi svého produktu readyBase, dynamicky rozšiřitelné databázové knihovny s API pro C, Perl a Javu. Knihovna je k dispozici pro Linux (Intel), Solaris (SPARC, Intel) a IRIX. Třicetidenní demoverze je k mání na adrese (1).

Olaf Titz (olaf@bigred.inka.de) uvolnil další verzi balíku `cipe`, který umožňuje kryptovat IP tunnel. `cipe` funguje na principu posílání kryptovaných UDP paketů, ale bohužel není kompatibilní s IPSEC. Tohoto balíku můžete s výhodou použít při budování VPN *Virtual Private Networks*. Bližší informace naleznete na adrese (2).

Balík TEItools, jehož autorem je Boris Tobotras (tobotras@jet.msk.su), zjednodušuje práci se SGML dokumenty a umožňuje z jednoho zdrojového textu generovat HTML, RTF, PostScript i PDF. Další informace naleznete na adrese (3).

7. dubna vyšlo tiskové prohlášení sdružení XFree86 ke změně licenčních podmínek referenční implementace (SI — *Sample Implementation*) standardu X11R6.4 společnosti TOG (*The Open Group*). Kompletní zprávu si můžete přečíst na adrese (4).

Pokud postrádáte na svém Linuxu BASIC, nezoufejte a podívejte se na adresu (5). Naleznete zde PC Linux BASIC, dobře dokumentovaný (manuál má 76 stránek) překladač jazyka BASIC. Autorem překladače je Mike Maddock (mike@lostboy.demon.co.uk).

1. dubna vyšlo již 27. číslo Linux Gazette (6). Kompletní číslo je k dispozici na adrese (7).

H. J. Lu (hjl@lucon.org) oznámil vytvoření binárek překladače `egcs` verze 1.0.2. Více informací získáte na adrese (8).

Owen Taylor (otaylor@gtk.org) oznámil 16. dubna, že je na světě dlouho očekávaná verze 1.0.0 knihovny `gtk+`. Kompletní informace naleznete na adrese (9).

22. dubna byla uvolněna další betaverze (tentokráte už čtvrtá) KDE (10) (*The K Desktop Environment*).

Andries Brouwer (Andries.Brouwer@cwi.nl) zveřejnil další verzi prohlížeče manuálových stránek — programu `man`. Verzi 1.5c naleznete na adrese (11). Andries také den poté uvolnil další verzi balíku manuálových stránek — `man-pages-1.19`, kterou naleznete na adrese (12). ■

Linuxové noviny připravuje a vydává České sdružení uživatelů operačního systému Linux

1 readyBase demo
http://www.dynamical-systems.com
2 CIPE 0.5.6
ftp://ftp.inka.de/sites/bigred/sw
3 TEItools
http://xtalk.price.ru/SGML/TEItools
4 XFree86
http://www.xfree86.org/
5 Linux BASIC
ftp://ftp.demon.co.uk/pub/unix/linux/apps/misc
6 Linux Gazette
http://www.linuxgazette.com/
7 Linux Gazette FTP server
ftp://ftp.ssc.com/pub/lg/
8 Egcs 1.0.2
http://egcs.cygnus.com
9 Gtk+
http://www.gtk.org/
10 The K Desktop Environment
http://www.kde.org
11 Man 1.5c
ftp://ftp.win.tue.nl/pub/linux/util
12 Man-pages 1.19
ftp://ftp.win.tue.nl/pub/linux/man

Linux Journal

Pavel Janík ml., 5. května 1998

Dubnové číslo časopisu Linux Journal (1), který je vydáván vydavatelstvím Specialized Systems Consultants, Inc. (SSC) (2), je zaměřeno na různá uplatnění operačního systému Linux v praxi.

Cliff Seruntine nám představí Linux z jiné stánky, než jej známe. Seznámí nás s možnostmi jak a proč vlastně Linux prodávat. Dalším článkem, který se tematicky zabývá nasazením Linuxu v praxi, je článek Rogera S. Flugela o Linuxu jako výpočetním systému v lékařské laboratoři. Ale Linux se nepoužívá jen v akademické oblasti. Článek Dmitrie Komarova o Linuxu ve vládních úřadech Litvy je toho důkazem. Autor používá hlavně `dosemu` a `mars_nwe` pro emulaci NetWare serveru.

> I've found running the application from dosemu on Linux is more stable than running it from Windows 3.11.
>
> ---
>
> Spouštění aplikací z dosemu je stabilnější než jejich spouštění pod Windows 3.11

Jon Davis a jeho kolegové používají Linux pro ovládání hardwarového vybavení. Dokonce si pro své experimenty vytvořili i vlastni integrované vývojové prostředí.

Z recenzí, které jsou v dubnovém čísle Linux Journalu, se mi jako nejzajímavější jeví článek o textovém procesoru WordPerfect 7.0. Další z recenzí je zaměřena na programátorský editor Visual SlickEdit, který umožňuje spoustu věcí, podobně jako např. editor Emacs.

Jedním z prvních komerčních programů pro Emacs je spell checker TeraSpell 97, jehož recenzi si můžete v dubnovém čísle také přečíst.

Pravděpodobně nejzajímavějším článkem (alespoň pro mne) je úvod do psaní driverů v Linuxu. Autorem článku je Fernando Matía. Jedinou stinnou stránkou je ovšem použitá verze jádra, na které jsou všechny příklady ukazovány (2.0.24).

Další článek mne poněkud zaskočil. Kdybych jej totiž četl o jeden či dva dny dříve, jistě by se hodnocení mé písemné práce z předmětu „Firemní finance" poněkud zvýšilo… James Shapiro mi totiž ukázal, jak v Perlu, v C a Javě napsat program, který počítá *Internal Rate of Return*, tedy *vnitřní výnosové procento*. Jedná se o jednoduchý problém, na kterém autor názorně demonstruje výhody a nevýhody jednotlivých jazyků.

Mezi dalšími články nalezneme úvod do programování CORBA (*Common Object Request Broker Architecture*), interview s viceprezidentem společnosti Corel Corporation, úvod do používání příkazu `imake` apod.

Velice zajímavý je také článek Erica S. Raymonda o uvolnění zdrojových textů prohlížeče Netscape a o free softwaru (resp. Open Source) vůbec.

V Linux Journalu najdeme navíc i recenze dvou nových knih nakladatelství Prentice Hall — *Protecting Your Web Site with Firewalls* a *Practical Programming in Tcl&Tk*. ■

1 Linux Journal
http://www.linuxjournal.com
2 Specialized Systems Consultants, Inc.
http://www.ssc.com

Co se stane, když

Pavel Janík ml., 2. května 1998

Měsíc uplynul jako voda a máme tu další soutěž v Linuxových novinách. Ta minulá byla poněkud techničtěji zaměřena, a proto bude nová soutěž lehčího charakteru, aby se mohli zúčastnit všichni. Opět se bude hrát o knihu věnovanou nakladatelstvím Neokortex, spol. s r. o. (1). Tentokráte o knihu *Apache a PHP/FI 2.0*, jejíž recenzi jste si mohli přečíst v minulém čísle Linuxových novin.

A nyní k vlastní soutěži. Před chvílí jsem si četl zajímavou literární úvahu na téma „Co se stane, když", která skončila tím, že si vlastně tuto otázku nemáme pokládat… Tato úvaha mne inspirovala k tomu, abych si položil následující otázku: „Co se stane, když společnost Microsoft uvolní zdrojové kódy operačního systému Windows?". A protože jsem nenalezl uspokojivou odpověď, tak tutéž otázku pokládám vám, čtenářům Linuxových novin. Vaše odpovědi očekávám jako vždy na adrese (2) se subjectem „LN: Soutez" (bez uvozovek). Uzávěrka soutěže je opět v den uzávěrky dalšího čísla Linuxových novin, tedy 9. června tohoto roku, a výsledky budou oznámeny v červnovém čísle Linuxových novin.

Nejvtipnější odpovědi budou zveřejněny v rubrice „Zasmáli jsme se" a ta opravdu jedinečná bude oceněna zmíněnou knihou. ■

1 Neokortex, spol. s r. o.
http://www.neo.cz
2 Redakce Linuxových novin
mailto:noviny@linux.cz

```
#define _syscall1(type,name,type1,arg1) \
   type name(type1 arg1) \
   { \
      long __res; \
      __asm__ volatile ("int $0x80" \
                        : "=a" (__res) \
                        : "0" (__NR_##name),"b" ((long)(arg1))); \
      if (__res >= 0) \
         return (type) __res; \
      errno = -__res; \
      return -1; \
   }
```

Výpis č. 1: Volání jádra na procesorech Intel

Jak jsem pěstoval trpaslíky

Pavel Kaňkovský, 7. května 1998

Slovo na úvod

V dubnovém čísle Linuxových novin byla vyhlášena soutěž o nejmenší soubor spustitelný pod Linuxem. Když jsem se dozvěděl, že program může dělat úplně cokoli, je-li to dobře definováno, zaujalo mne to a také jsem se zúčastnil.

Smršťování kódu

Nejdříve jsem si položil otázku: *Jaký je nejmenší možný kód (bez hlaviček a podobného smetí), který ještě pod Linuxem dělá něco smysluplného?* Je zřejmé, že takový program napsaný v jazyce C by vypadal například takto:

```
int main()
{
   return 0;
}
```

to jest nedělá nic jiného, než že skončí.

Ovšem takový program je ve skutečnosti obalen kódem knihovny, která po návratu z funkce `main()` volá funkci `exit()`, která pak volá stejnojmené systémové volání, které ukončí život procesu. Což znamená, že program je možno „zjednodušit“ takto (případné varování překladače, že nezná návratovou hodnotu, budeme ignorovat):

```
#include <syscall.h>
int main()
{
   syscall(SYS_exit, 0);
}
```

Nyní musíme volání funkce (resp. makra) `syscall()` „rozložit na prvočinitele“. Nahlédneme do souboru `/usr/include/asm/unistd.h` a podíváme se, jak se provádí volání jádra na procesorech Intel (viz výpis Volání jádra na procesorech Intel).

Poněkud kryptické, není-liž pravda? Nicméně cvičené oko odhalí, že je vhodné nastavit registr `eax` na číslo systémového volání (v našem případě je to číslo 1), registr `ebx` nastavit na hodnotu argumentu tohoto volání (tedy nulu) a vyvolat přerušení číslo 0x80.

Nyní již opusťme (relativně) pohostinné končiny jazyka C a přepišme celý program do assembleru (neboli „jazyka symbolických adres“ pro přátele starých pořádků):

```
.text
.globl _start
_start:
   mov     $1, %eax
   mov     $0, %ebx
   int     $0x80
```

Takový program se už obejde bez podpory podobných zbytečností, jako jsou knihovny, takže už nebudeme ani předstírat nějakou funkci `main()` a rovnou definujeme symbol `_start` (a ušetříme si jeden parametr při volání linkeru, neboť `_start` je implicitní název vstupního bodu programu, který se obyčejně nachází v `/usr/lib/crt1.o`).

Náš zdrojový text nazveme například `program.s` a přeložíme příkazem `cc -c program.s`. Výsledkem bude následujících 12 bajtů strojového kódu (`objdump --disassemble --show-raw-insn program.o`):

```
00000000 <_start>   b8 01 00 00 00  movl $0x1,%eax
00000005 <_start+5> bb 00 00 00 00  movl $0x0,%ebx
0000000a <_start+a> cd 80           int  $0x80
```

Je vidět, že značné rezervy jsou v nastavení obou registrů: 10 bajtů se zdá být značným plýtváním prostorem. Assemblerový expert by kód upravil například takto:

```
.text
.globl _start
_start:
   xor     %eax, %eax
   mov     %eax, %ebx
   inc     %eax
   int     $0x80
```

což vytvoří následujících 7 bajtů kódu:

```
00000000 <_start>   31 c0  xorl %eax,%eax
00000002 <_start+2> 89 c3  movl %eax,%ebx
00000004 <_start+4> 40     incl %eax
00000005 <_start+5> cd 80  int  $0x80
```

Ale to ještě není konec! Ukazuje se (a lze to ověřit nahlédnutím do zdrojových textů jádra), že registr `eax` je již při vstupu do programu vynulován — jako by docházelo k úspěšnému návratu ze systémového volání `execve()`. Proto lze vynechat první instrukci. Nutným předpokladem ovšem je, že náš kód je skutečně to první, co bude vykonáno, nikoli dynamický linker nebo něco podobného, proto je žádoucí při linkování používat parametr `-static`.

Nyní lze přistoupit k zřejmě již definitivně poslední optimalizaci: nastavení registru `ebx` spotřebovává dva bajty. Tomu se jen těžko lze vyhnout, má-li mít nulovou (nebo

jedničkovou) hodnotu, nicméně zadání vyžaduje, aby měl hodnotu „dobře definovanou“. Iniciální hodnota při spuštění tuto podmínku bohužel — narozdíl od registru `eax` — nesplňuje. Naštěstí je jedna vhodná hodnota hned po ruce: jedná se o počet argumentů programu (známý pod přezdívkou `argc`), který je jádrem uložen na samotný vrchol zásobníku. Náš program je tedy možné zkrátit na tři instrukce zvíci pouhých **čtyř** bajtů:

```
00000000 <_start>   40     incl   %eax
00000001 <_start+1> 5b     popl   %ebx
00000002 <_start+2> cd 80  int    $0x80
```

neboli ve zdrojové formě (která je teď pro změnu uvedena jako druhá):

```
.text
.globl _start
_start:
   inc     %eax
   pop     %ebx
   int     $0x80
```

což je víceméně ekvivalent tohoto céčkového programu:

```
int main(int argc)
{
   return argc;
}
```

Tento kód již vypadá celkem minimálně, takže z něj zkusíme vytvořit spustitelný soubor:

```
$ cc -c program.s
$ ld -static program.o
$ ls -l a.out
-rwxr-xr-x 1 peak users 675 May 3 22:00 a.out
```

Ó jé! Takový malý kód a tak velký soubor! S tím se musí něco udělat.

Smršťování souboru

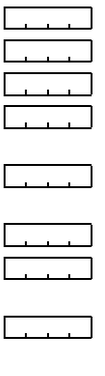
Nyní je třeba řešit další zásadní problém: *Jak vyrobit co nejmenší spustitelný soubor?*

Linux zná několik druhů spustitelných souborů: základní jsou **a.out**, **ELF** a `skripty` (tj. soubory začínající znaky `#!`). Skripty vynecháme, protože nevyhovují původnímu zadání.

- **a.out** je starší a jednodušší formát. Jeho kořeny sahají někam do šerého unixového dávnověku, ale v dnešní době se už prakticky nepoužívá. Má několik variant, které se liší způsobem načítání do paměti. Viz `include/{linux,asm}/a.out.h` a `fs/binfmt_aout.c`.

- **ELF** je novější a složitější formát. Byl vyvinut pro potřeby System V. Ve své úplnosti je tak složitý, že pro jeho dekódování existuje zvláštní knihovna (`libelf`), ale nás zajímá jen malá část — přesně ta, co zajímá jádro. Viz `include/{linux,asm}/elf.h` a `fs/binfmt_elf.c`.

Věnujme se nejprve krátce formátu **a.out**. Jeho hlavička vypadá takto:



`a_info`	„magické číslo“
`a_text`	délka kódu
`a_data`	délka inicializovaných dat
`a_bss`	délka neinicializovaných dat
`a_syms`	velikost tabulky symbolů v bajtech
`a_entry`	startovací adresa
`a_trsize`	velikost relokační tabulky pro kód
`a_drsize`	velikost relokační tabulky pro data

Spustitelný soubor je tvořen hlavičkou a za sebou následujícími úseky kódu, inicializovaných dat a tabulky symbolů (neinicializovaná data nejsou pochopitelně v souboru obsažena). Soubor **nesmí** obsahovat žádná relokační data a příslušné hodnoty v hlavičce musí být nulové.

„Magické číslo“ popisuje variantu formátu. Nabývá jedné z následujících hodnot:

`OMAGIC`	tzv. „impure executable“; zbytek souboru je načten do paměti od adresy 0, zápis je povolen všude (pozn. tato varianta bývala používána i pro objektové soubory formátu **a.out**(`*.o`)
`NMAGIC`	tzv. „pure executable“; zbytek souboru je načten do paměti od adresy 0, ale do oblasti kódu je zakázán zápis
`ZMAGIC`	tzv. „demand-paged executable“; soubor počínaje pozicí 4096 je namapován do paměti od adresy 0x1000, v prostoru první stránky není nic namapováno
`QMAGIC`	jiná varianta „demand-paged executable“, celý soubor je namapován do paměti od adresy 0x1000, narozdíl od předchozí varianty není na začátku souboru jedna prakticky nevyužitá stránka

Zkusíme vytvořit soubor ve formátu **a.out** pomocí standardních nástrojů. Jediný problém je v tom, že musíme linkeru dát parametr `-m i386linux` nebo `-b a.out-i386-linux`, protože (pokud není provozovaná verze opravdu letitá) bez těchto parametrů by byl vytvořen soubor ve formátu **ELF**. Parametrem `-N` si vyžádáme vytvoření varianty `OMAGIC` (i když stejně dobře by posloužila varianta `NMAGIC` vyvolaná parametrem `-n`). Navíc (narozdíl od příkladu v předchozí kapitole) nezapomeneme odstranit ladící informace pomocí příkazu `strip` (což je kupodivu účinnější než použití parametru `-s` při linkování).

```
$ ld -static -m i386linux -N program.o
$ strip a.out
$ ls -l a.out
-rwxr-xr-x 1 peak users 40 May 3 22:05 a.out
```

Soubor obsahuje 32 bajtů hlavičky a 4 bajty kódu zvětšené o 4 bajty výplně. To je docela pěkný výsledek, ale, jak uvidíme záhy, není zdaleka nejlepší možný.

Přikročme nyní ke studiu formátu **ELF**. Ten má dvě zajímavé hlavičky. První je hlavička souboru známá pod názvem `Elf32_Ehdr` (existuje i 64-bitová varianta `Elf64_Ehdr`):

	e_ident	identifikace souboru
		…
		…
		…
	e_type	typ souboru
	e_machine	platforma
	e_version	verze formátu souboru
	e_entry	startovací adresa
	e_phoff	začátek tabulky segmentů (v souboru)
	e_shoff	začátek tabulky sekcí (v souboru)
	e_flags	různé příznaky
	e_ehsize	velikost hlavičky souboru v bajtech
	e_phentsize	velikost záznamu segmentu
	e_phnum	počet záznamů v tabulce segmentů
	e_shentsize	velikost záznamu sekce
	e_shnum	počet záznamů v tabulce sekcí
	e_shstrndx	index sekce obsahující jména

a druhá je hlavička segmentu (tedy položka v tabulce segmentů) čili `Elf32_Phdr`:

	p_type	druh segmentu
	p_offset	počátek segmentu v souboru
	p_vaddr	virtuální adresa v paměti
	p_paddr	fyzická adresa v paměti
	p_filesz	velikost segmentu v souboru
	p_memsz	velikost segmentu v paměti
	p_flags	příznaky segmentu
	p_align	zarovnání adresy segmentu

Spustitelný soubor ve formátu **ELF** je tvořen hlavičkou souboru a „volným prostorem“, ve kterém jsou víceméně libovolně rozloženy ostatní části souboru: tabulky segmentů a sekcí a samotné segmenty a sekce. Tyto části se mohou libovolně překrývat (nenaruší-li to integritu jejich obsahu).

První položkou v hlavičce souboru je identifikace: 16 bajtů, které určují, že se jedná o soubor ve formátu **ELF** a popisuje jeho základní vlastnosti (LSB/MSB, 32/64 bitů, verzi formátu hlavičky souboru). Celkem je obsazeno pouze 7 bajtů, zbylých 9 by mělo mít nulovou hodnotu, ale nutné to není. Později toho bude využito.

Položka `e_type` určuje typ souboru, spustitelné soubory zde mají hodnotu 2, symbolicky `ET_EXEC`. Položka `e_machine` určuje hardwarovou platformu, pro kterou je soubor určen — v našem případě to bude 3, symbolicky `EM_I386`. Položka `e_version` určuje verzi formátu, která zatím existuje pouze jediná s číslem 1.

Důležité položky hlavičky souboru jsou ještě `e_entry`, která určuje adresu, od které bude náš program vykonáván, `e_phoff` obsahující polohu tabulky segmentů v souboru, `e_phnum` udávající počet segmentů a `e_phentsize`, která musí obsahovat velikost hlavičky segmentu, v bajtech, tedy číslo 32.

Zbylé položky nejsou při načítání souboru do paměti jádrem nijak interpretovány, takže mohou mít prakticky libovolné hodnoty.

Segment je ta část souboru, která má být při spouštění programu jádrem načtena či namapována do paměti, tedy kód nebo data. Existují sice také některé druhy segmentů mající speciální význam, ale my budeme blíže zkoumat pouze právě popsaný typ `PT_LOAD`, číselně se jedná o typ 1.

V hlavičce segmentu je (narozdíl od hlavičky souboru) důležitá většina údajů s výjimkou `p_paddr` (neboť mapování na konkrétní fyzické adresy není možné) a `p_align` (segmenty jsou vždy zarovnány na celé stránky). Zvláštní pozornost zaslouží `p_flags`, který popisuje, jestli bude možno namapovaný segment číst (`PF_R`, 4), vykonávat (`PF_X`, 1), resp. zda do něj bude možno zapisovat (`PF_W`, 2).

Zkusme nyní vyrobit co nejmenší spustitelný soubor ve formátu **ELF**. Nejprve použijeme linker a `strip`, jako v případě formátu **a.out**:

```
$ ld -static program.o
$ strip a.out
$ ls -l a.out
-rwxr-xr-x 1 peak users 364 May 3 22:23 a.out
```

Výsledek je to rozhodně lepší než předtím, ale pořád to není ono. Přeložený soubor ve skutečnosti obsahuje velké množství smetí, jak nám ukáže program `objdump`:

```
$ objdump -h a.out
a.out:     file format elf32-i386
Sections:
Idx Name Size     VMA      LMA      File off Algn
 0 .text 00000004 08048074 08048074 00000074 2**2
   CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .data 00000000 08049078 08049078 00000078 2**2
   CONTENTS, ALLOC, LOAD, DATA
 2 .bss  00000000 08049078 08049078 00000078 2**2
    ALLOC
```

V souboru jsou obsaženy tři sekce, ale jediná potřebná z nich je sekce `.text`, která obsahuje spustitelný kód. Pokusíme se soubor zmenšit odstraněním ostatních sekcí. K tomu použijeme program `objcopy`:

```
$ objcopy -R .data -R .bss a.out
$ ls -l a.out
-rwxr-xr-x 1 peak users  276 May 3 22:29 a.out
```

Soubor obsahuje 52 bajtů hlavičky souboru, jednu hlavičku segmentu o velikosti 32 bajtů, pak 32 bajtů nul a 4 bajty kód, dohromady jeden segment o velikosti 120 bajtů, po kterém pak následuje 156 bajtů s tabulkou sekcí a tabulkou jmen, což jsou ovšem věci pro vlastní chod programu celkem nepodstatné (je-li linkován staticky, u dynamicky linkovaných programů je to trochu jinak).

O mnoho více již s pomocí standardních nástrojů nelze dokázat. Nejjednodušší bude asi vyrobit celý soubor ručně…

Bitové inženýrství

Zkusme si tedy vyrobit nějaký spustitelný soubor sami — bajt po bajtu. Pro tento účel je vhodné použít nějaký „překladač“, který nám umožní zapsat binární data v nějaké čitelné textové podobě. Já jsem si jednu takovou jednoduchou pomůcku pro tento účel vyrobil a nazval jsem ji `hex`. Jazyk překladače `hex` je opravdu jednoduchý:

- vše počínaje znakem „#“ je až do konce řádku ignorováno
- obsah souboru je interpretován po slovech oddělených bílými znaky
- slovo obsahující „L“ je interpretováno jako 4-bajtová hodnota (*long*), slovo obsahující „W“ jako 2-bajtová (*word*), vše ostatní je jednobajtová hodnota; vícebajtové hodnoty jsou ukládány od nejméně významného bajtu k nejvýznamnějšímu (*little-endian*)
- slovo obsahující „x“ je interpretováno v šestnáctkové soustavě, slovo obsahující „o“ v osmičkové soustavě, ostatní v desítkové

Začneme opět s formátem **a.out** (viz výpis Zdrojový text pro a.out v jazyce hex), neboť je významně jednodušší.

```
# a.out hlavicka
Lx00640107 # a_info (OMAGIC)
L4  # a_text
L0  # a_data
L0  # a_bss
L0  # a_syms
L0  # a_entry
L0  # a_trsize
L0  # a_drsize
# kod
x40 # inc %eax
x5b # pop %ebx
xcd # int
x80 # $0x80
```

Výpis č. 2: Zdrojový text pro a.out v jazyce hex

Tento „zdrojový soubor“ se po průchodu překladačem `hex` změní na 36 bajtový soubor, který je jádro Linuxu ochotno načíst a spustit. Uspořili jsme zatím čtyři bajty výplně. Někdo může namítnout, že vhodnější volbou parametrů překladu bychom mohli dosáhnout téhož. Ale ještě jsme neskončili!

Všimněme si nyní dvou věcí: za prvé, pokud změníme typ souboru na `QMAGIC`, bude do paměti načten — resp. namapován — celý soubor (i když jádro bude poněkud „nervózní“ z toho, že délka souboru není násobek 4096 bajtů), za druhé, pokud vynecháme koncovou část hlavičky, domyslí si na tom místě jádro nuly (těžko říci, zda je to chyba nebo úmysl). Náš kód má pouze čtyři bajty, takže by ho bylo možno vecpat do některé položky v hlavičce a ještě ji na konci uříznout. Ač to zní překvapivě, možné to je: kód lze vložit do položky `a_bss`, ačkoli podle zdravého rozumu by něco takového nemělo jádro vůbec spustit (velikost neinicializovaných dat je větší než 2GB). Spustí…

Výsledný program vypadá tak, jak je zobrazeno na výpise Finální zdrojový text pro formát a.out.

Hrůza! Ale na druhou stranu má výsledný spustitelný soubor velikost pouhých 24 bajtů (tedy o 8 méně než je regulérní velikost hlavičky). Menší spustitelný soubor už asi nikdo nevyrobí. Škoda jen, že při každém spuštění jádro vypíše `executable not page aligned`.

A co formát **ELF**? Začneme opět konzervativně (viz výpis Zdrojový text pro ELF v jazyce hex).

Nejprve 52 bajtů hlavičky souboru, pak 32 bajtů hlavičky segmentu a pak čtyři bajty kódu. Dohromady 88 bajtů.

Zkusme nyní ušetřit nějaké místo. Prvním krokem bude asi využití prázdného místa v `e_ident`: nejjednodušší

```
# a.out hlavicka
Lx006400cc # a_info (QMAGIC)
L20 # a_text
L0  # a_data
# L0  # a_bss
# kod (na miste a_bss!)
x40 # inc %eax
x5b # pop %ebx
xcd # int
x80 # $0x80
L0  # a_syms
Lx100c  # a_entry
# ty tady nemusi byt, protoze jsou nulove
# (poznamenejme, ze nulove byt MUSI)
# L0  # a_trsize
# L0  # a_drsize
```

Výpis č. 3: Finální zdrojový text pro formát a.out

```
# Elf32_Ehdr
# e_ident
x7f x45 x4c x46 # \177ELF
x01 x01 x01 x00 # 32bit, LSB, current version
L0
L0
# dalsi polozky
W2      # e_type = ET_EXEC
W3      # e_machine = EM_386
L1      # e_version
Lx1054  # e_entry
Lx34    # e_phoff
L0      # e_shoff
L0      # e_flags
Wx34    # e_ehsize
Wx20    # e_phentsize
W1      # e_phnum
Wx28    # e_shentsize
W0      # e_shnum
W0      # e_shstrnum
# Elf34_Phdr
L1      # p_type = PT_LOAD
L0      # p_offset
Lx1000  # p_vaddr
Lx1000  # p_paddr
L88     # p_filesz
L88     # p_memsz
L5      # p_flags = PF_R | PF_X
Lx1000  # p_align
# kod
x40 # inc %eax
x5b # pop %ebx
xcd # int
x80 # $0x80
```

Výpis č. 4: Zdrojový text pro ELF v jazyce hex

bude do něj přesunout kód — tím soubor zkrátíme o čtyři bajty. Lepší by bylo někam přesunout hlavičku segmentu, ale ta je moc velká, do `e_ident` se rozhodně nevejde, ani kdybychom odtud odstranili kód, který jsme tam zrovna umístili… Ovšem pozor! V hlavičce souboru je spousta zbytečného místa: umístíme-li začátek hlavičky segmentu na stejné místo jako `e_shoff`, dojde k jediné významné kolizi: druhý bajt `p_vaddr` bude na stejném místě jako `e_phentsize`, což je jednoduché vyřešit — stačí vhodně

zvolit adresu, na kterou program v paměti umístíme. Tím ušetříme 20 bajtů, další čtyři ještě můžeme získat useknutím položky `e_align`, kterou jádro postrádat nebude. Program pak bude vypadat tak, jak je na výpisu Finální zdrojový text pro formát ELF.

```
# Elf32_Ehdr
# e_ident
x7f x45 x4c x46 # \177ELF
x01 x01 x01 x00 # 32bit, LSB, current version
# kod (pres zbytek e_ident)
x40 # inc %eax
x5b # pop %ebx
xcd # int
x80 # $0x80
L0      # tady je jeste mista....
# dalsi polozky
W2      # e_type = ET_EXEC
W3      # e_machine = EM_386
L1      # e_version
Lx200008# e_entry
Lx20    # e_phoff
#                       Elf34_Phdr
L1      # e_shoff       p_type = PT_LOAD
L0      # e_flags       p_offset
W0      # e_ehsize      p_vaddr = 0x00200000
Wx20    # e_phentsize (!)
W1      # e_phnum       p_paddr = 0x00280001
Wx28    # e_shentsize
Lx20    # e_shnum       p_filesz
        # e_shstrnum
Lx20    #               p_memsz
L5      #               p_flags = PF_R | PF_X
        #               p_align (vynechano)
```

Výpis č. 5: Finální zdrojový text pro formát ELF

Celková velikost je neuvěřitelných 60 bajtů! Navíc takový program funguje, jádro si na nic nestěžuje a program `file` ho správně identifikuje:

```
$ ls -l elf_tiny
-rwxr-xr-x 1 peak users  60 May 3 22:47 elf_tiny
$ file elf_tiny
elf_tiny: ELF 32-bit LSB executable, \
Intel 80386, version 1,
statically linked, stripped
$ ./elf_tiny 1 2 3; echo $?
4
```

A to je vše, přátelé! Menší programy jsem už opravdu vyrobit nedokázal. ■

IPX v Linuxu

Milan Keršláger, 11. května 1998

Úvod do IPX/SPX

S protokolem IPX/SPX se setkáváme denně, zejména v prostředí Novell NetWare, kde je klíčovým protokolem. Protokol IPX/SPX počítá s vysokou rychlostí a nízkou chybovostí lokálních sítí a je jednodušší než protokol TCP/IP, který je používán v rozlehlém světě Internetu. To je hlavní důvod, proč je v lokálních sítích tolik používán. Jeho konfigurace i implementace je jednoduchá a neskrývá mnoho úskalí. Protokol IPX (*Internetwork Packet eXchange*) pracuje jako datagramová služba a je tak obdobou protokolu IP. Naproti tomu SPX (*Sequenced Packet eXchange*) je navazováním spojení velice blízký protokolu TCP.

K čemu to bude dobré

Všechny počítače v síti jsou si při vzájemné komunikaci rovny. Přesto mají některé významnější postavení než ostatní. Jsou to většinou specializované stanice, které nabízejí své služby ostatním stanicím. Takovým počítačům říkáme *servery*, stanicím pak *klienti* a říkáme, že používáme architekturu *klient–server*. Určitě je velmi výhodné umět i z Linuxu využívat všech služeb, které nabízejí servery Novell Netware. Abychom mohli s těmito servery komunikovat, musíme umět nakonfigurovat svůj Linux tak, aby jim rozuměl. V dalším povídání budu přepokládat, že máme k dispozici síť typu Ethernet (např. tenký koaxiál nebo kroucenou dvoulinku), a podíváme se blíže, jak to všechno zařídit. Začneme od začátku, tedy od toho, jak komunikace na Ethernetových sítích vlastně funguje.

Nejprve se pokusím objasnit význam pojmů datagram, paket a rámec. *Datagram* je celistvá informace, kterou přenášíme mezi počítači. Datagram může být při přepravě dělen na více částí a takovýmto samostatným částem říkáme *pakety*. Aby mohl být paket přepravován po síti typu Ethernet, je ho potřeba doplnit o další nezbytné informace nutné pro jeho přepravu po síťovém médiu, a takovémuto rozšířeném paketu říkáme *rámec*.

Ethernetové rámce

Základem přepravy dat na sítích typu Ethernet jsou takzvané rámce, které přepravují obecně jakákoliv data mezi jednotlivými počítači. Každý rámec představuje balík dat, který je možno vyslat v rámci jednoho segmentu k jinému počítači. Skládá se z hlavičky, těla (obsahuje přenášená data, tedy paket vyšší vrstvy) a traileru (zakončení rámce s kontrolním součtem). Hlavní část hlavičky tvoří hardwarová adresa odesílatele a příjemce (označovaná také jako fyzická MAC adresa), podle které síťové rozhraní dokáže samo rozeznat rámce, které jsou mu určeny a nemusí tak rušit počítač zbytečnými přerušeními. MAC adresa je pevně dána výrobcem síťové karty a většinou ji není možné měnit. Teoreticky by se na světě neměly vyskytovat dvě Ethernetové karty se stejnou MAC adresou, ale může se to stát, i když výbor IEEE (Institute of Electrical and Electronics Engineers) přiděluje každému výrobci blok adres. Pokud na takové karty narazíte, je to nepříjemné a máte z pekla štěstí :-).

Rámec je závislý na použitém přepravním médiu. Když datagram opouští Ethernet a je dále přepravován například po sériové lince, jsou zachována pouze data z těla rámce a ta pak putují dále obalena jiným přepravním protokolem.

Abychom neměli pocit, že je všechno jednoduché, existuje více norem, které předepisují, jak má hlavička rámce v sítích typu Ethernet vypadat (viz tabulka Ethernetové rámce). Síťová karta musí umět správně dekódovat přijatý rámec, jinak ho může považovat za chybný. Proto je vhodné používat na segmentu pokud možno jen jeden rámec a nepřivádět jak síťovou kartu tak administrátora do schizofrenického stavu. Otázkou pak ovšem zůstává, který rámec je ten pravý.

Název rámce	Charakteristika
Ethernet 802.2	dnešní standard firmy Novell, umí přepravovat pouze IPX/SPX
Ethernet 802.3	původní standard firmy Novell, umí přepravovat pouze IPX/SPX
Ethernet II	umí přepravovat IPX/SPX i TCP/IP, nejrozšířenější, nejjednodušší
Ethernet SNAP	umí přepravovat IPX/SPX i TCP/IP

Tabulka č. 1: Ethernetové rámce

Pokud chcete používat ve své síti protokol TCP/IP, nevyhnete se použití jednoho z posledních dvou rámců. Nejvýhodnějším je určitě Ethernet II, protože je nejjednodušší ze všech a hlavně je univerzální (lze v něm přepravovat kromě IPX/SPX a TCP/IP spoustu dalších protokolů). O rámci 802.2 se někdy říká, že je přenášen v rámci 802.3 (při použití paketového analyzátoru se tak jeho struktury jeví). Ethernet SNAP je kombinací rámců 802.2 a 802.3. Rozumný důvod, proč firma Novell tak vehementně prosazuje právě první dva, neznám.

Někdy není možné na jednom segmentu vystačit jen s jedním rámcem, protože například některé BootROMky nebo stanice se staršími ovladači rámec Ethernet II neumějí. Ovšem určitě platí, že v jednoduchosti je síla. Proto raději důkladně zvažte, co všechno Vám na segmentu bude běhat. Neexistuje žádný důvod, proč by neměl být rámec Ethernet II používán.

Osobně používám na segmentech jen jeden rámec (Ethernet II) a ostatní přidávám jen v případě nezbytné nutnosti. Windows 95 není vhodné nechávat u protokolu IPX/SPX používat autodetekci rámců, protože to občas vede k nepochopitelným výpadkům, zvláště ve větších sítích.

Dělení na více sítí

Protože rámce neobsahují možnost, jak je jednoduše směrovat mezi více segmenty (ani to není jejich úkol), je směrování zajištěno protokolem vyšší vrstvy, v našem případě tedy protokolem IPX. Adresa se v IPX datagramu skládá z čísla sítě a čísla uzlu, které kopíruje uz výše zmíněnou MAC adresu síťové karty. Každý segment musí mít jiné číslo sítě a pokud provozujeme na jednom segmentu více rámců, pak i každý rámec musí mít různá čísla sítí. Všechna síťová rozhraní na jednom segmentu se stejným rámcem používají stejná čísla sítí. Ta určuje administrátor při instalaci serverů a pokud dojde ke konfliktu, v horším případě to vyřadí z provozu celý segment. Stanice si obvykle zjistí číslo sítě ze serveru sama, bez nutnosti přesné konfigurace.

Vnitřní sítě

Vnitřní sítě (tzv. *internal network*) slouží pro snadné směrování datagramů ze stanic na místo, které Vám poskytuje nějakou službu. Vnitřní číslo sítě proto potřebuje jen server, stanice bez něj bude pracovat a vlastně ho ani na nic nepotřebuje. Vnitřní číslo sítě definuje jakousi virtuální síť uvnitř serveru, na které sice není určen žádný rámec, ale přesto musí být její číslo jedinečné a nesmí se tedy shodovat s jiným číslem sítě v naší lokální síti.

Spolupráce Linuxu s NetWare servery

Servery Novell NetWare nejčastěji komunikují se stanicemi pomocí protokolu IPX, ten je ovšem pouze přepravním protokolem. Dnes můžeme jako přepravní protokol využít také TCP/IP a překročit tak mnohem větší vzdálenosti než s protokolem IPX, ovšem konfigurace síťových rozhraní u stanic je pak výrazně složitější. Navíc je podpora TCP/IP pro NetWare na straně Linuxu zatím v plenkách. Proto se dále budeme zabývat pouze první a zřejmě také nejrozšířenější variantou.

Protokoly v prostředí NetWare

Aby si stanice se servery při vzájemném rozhovoru rozuměly, přepravovaná data měla logiku a strukturu, potřebujeme další protokoly, které budou v IPX přepravovány. Nejvýznamějším takovým protokolem je NCP (*NetWare Core Protocol*), který umožňuje klientům přístup k síťovým diskům, tiskárnám a dalším sdíleným zařízením. Pomocí tohoto protokolu se uživatelé také k serverům NetWare přihlašují. Dalším protokolem, se kterým se setkáme, je SAP (*Service Advertisement Protocol*). Tímto protokolem jednotlivé servery ohlašují a nabízejí své služby v síti pomocí broadcastů. Posledním zajímavým protokolem je RIP (*Routing Information Protocol*), pomocí kterého si mezi sebou IPX směrovače (routery) vyměňují informace o známých sítích. Ty se pak odrazí ve směrovacích tabulkách a tím je umožněno, aby směrovače mohly předávat datagramy i do jiných sítí, než ve které pracuje stanice. Přes router může být zajištěn i překlad rámce, ale pokud se tak děje na jednom segmentu, povede to k duplikaci provozu. To je další důvod, proč je vhodnější používat na všech stanicích v síti pouze jeden jediný rámec.

Konfigurace IPX v Linuxu

Pokud má Linux vystupovat jako obyčejná stanice, která rozumí protokolu IPX, bude vystupovat jako klient a bude využívat zdroje a služby (např. sdílení disků a tisk na síťových tiskárnách), které mu budou pomocí sítě poskytovány servery.

Při konfiguraci musíme začít od základu a tím je protokol IPX. Podpora protokolu IPX musí být povolena při kompilaci jádra Linuxu (`CONFIG_IPX`). Pokud ji máme připravenu jako modul, musíme ho nejprve do jádra ručně zavést (neplatí pro nejnovější verzi NCP utilit). Pokud podporu IPX v jádru nemáme, nezbude nám nic jiného, než si přeložit jádro znovu. Pak je potřeba správně nakonfigurovat síťové rozhraní, k čemuž potřebujeme několik utilit, které najdeme v balíku `ncpfs`. V tomto balíku se nachází většina utilit, které pro práci s IPX budeme v Linuxu potřebovat. Uživatelům distribuce RedHat 5.0 stačí nainstalovat balík `ncpfs-2.0.11-3.i386.rpm` nebo podobný, IPX je v jádrech k dispozici jako modul.

Síťové rozhraní můžeme nechat nakonfigurovat automaticky, pokud je na síti už IPX používáno. Automatickou konfiguraci obstarává jádro odposlechem provozu na síti. Pokud na síti není žádný IPX provoz, musíme rozhraní nakonfigurovat ručně. Pro běžný provoz automatickou konfiguraci nedoporučuji, protože Windows 95 vysílají do sítě chybné pakety a to pak může vést k nesprávnému rozpoznání čísla sítě a používaných typů rámců, proto je vhodná spíše jen pro první kroky. Pro manuální konfiguraci potře-

bujete znát číslo sítě pro každý používaný rámec. Pokud ho nevíte, informujte se u svého administrátora.

Automatická konfigurace:

```
modprobe ipx
ipx_configure --auto_interface=on \
   --auto_primary=on
cat /proc/net/ipx_*
```

Automatickou konfiguraci je vhodné zkontrolovat podle výpisu souboru `/proc/net/ipx_interface`, který obsahuje seznam všech registrovaných IPX rozhraní včetně typů rámců a čísel sítí. Automatické nakonfigurování rozhraní může chvíli trvat (cca 10-30 vteřin).

K ruční konfiguraci IPX rozhraní můžeme použít utilitu `ipx_interface`. Právě jedno rozhraní by mělo být přepínačem `-p` označeno jako primární. Takové rozhraní je považováno za implicitní a je použito, když v programu při otevírání soketu neuvedeme číslo sítě.

```
ipx_interface činnost [-p] rozhraní\
              typ_rámce číslo_sítě
```

Činnost	Popis
add	přidává IPX rozhraní, vždy je nutné uvést název rozhraní a typ rámce. Pokud není uvedeno číslo sítě, je zjištěno odposlechem.
del	ruší na uvedeném rozhraní uvedený rámec
delall	ruší všechna rozhraní
check	zobrazí konfiguraci příslušného rozhraní

Typ rámce může být `802.2` pro rámec Ethernet 802.2, `802.3` pro rámec Ethernet 802.3, `EtherII` pro rámec Ethernet II a `Snap` pro rámec Ethernet SNAP.

Příklad ruční konfigurace rozhraní pro rozhraní `eth0`, rámec Ethernet II, číslo sítě 12 (primární interface) a rámec 802.3, číslo sítě 13:

```
modprobe ipx
ipx_interface add -p eth0 etherii 12
ipx_interface add    eth0 802.3   13
```

Po těchto krocích si můžeme ověřit konfiguraci například takto:

```
monkey:~# cat /proc/net/ipx_interface
Network  Node_Address Primary Device Frame_Type
00000012 00A024D6F9F9 Yes     eth0   EtherII
00000013 00A024D6F9F9 No      eth0   802.3
monkey:~# cat /proc/net/ipx_route
Network    Router_Net   Router_Node
00003333   00000012     0020AFF67402
00001111   00000013     0020AFF67402
00000013   Directly     Connected
00000012   Directly     Connected
monkey:~# ifconfig eth0
eth0
  Link encap:Ethernet  HWaddr 00:A0:24:D6:F9:F9
  inet addr:10.1.1.1  Bcast:10.1.1.255  \
    Mask:255.255.255.0
  IPX/Ethernet II addr:00000012:00A024D6F9F9
  IPX/Ethernet 802.3 addr:00000013:00A024D6F9F9
  UP BROADCAST RUNNING MULTICAST  MTU:1500  \
    Metric:1
  RX packets:587 errors:0 dropped:0 overruns:0
  TX packets:141 errors:0 dropped:0 overruns:0
  Interrupt:10 Base address:0x6100
```

Správnou funkci síťového rozhraní můžeme ověřit například výpisem dostupných NetWare serverů pomocí příkazu `slist`:

```
monkey:~> slist
Known NetWare File Servers Network   Node Address
---------------------------------------------------
PRUM                       0003333   000000000001
PAT                        0001111   000000000001
```

Pokud bychom chtěli nakonfigurovaná rozhraní z nějakého důvodu zrušit, mohli bychom postupovat například takto (na vašem systému nemusí být nutně použity všechny uvedené příkazy):

```
ipx_configure --auto_interface=off \
   --auto_primary=off
ipx_interface delall
rmmod ipx
```

Linux jako IPX směrovač

Pokud si to budeme přát, Linux může fungovat jako směrovač IPX protokolu mezi různými sítěmi. Směrování je prováděno na úrovni jádra operačního systému podle údajů ve směrovací tabulce IPX (vypsat si ji můžeme už zmíněným příkazem

```
cat /proc/net/ipx_route
```

Tuto činnost Linux vykonává podobně jako NW server a stejně jako on musí ohlašovat ostatním směrovačům a stanicím, jaké jsou dostupné sítě, a zároveň musí sám dle hlášení ostatních směrovačů automaticky upravovat své směrovací tabulky pro IPX protokol. O tuto činnost se stará zvláštní program.

Tento zvláštní program bývá označován jako démon a stačí ho vlastně jen spustit. O vše ostatní se postará sám, pokud před tím správně nakonfigurujeme všechna síťová rozhraní, což už ale perfektně umíme. Zajímavé je, že pokud budeme používat program Mars, o kterém ještě bude řeč, nebudeme démona potřebovat, protože je přímo součástí Marsu. To je velmi sympatické, i když možná někdy už méně praktické. ■

NetWare zdarma

Milan Keršláger, 11. května 1998

Novell NetWare

Novell NetWare (NW) je stabilní hvězdou na nebi lokálních sítí, kde umožňuje stanicím sdílet různá zařízení prostřednictvím počítačové sítě. Pracuje na základě architektury klient-server. Klienty zde jsou stanice, které sdílejí disky serveru, společné tiskárny, CD-ROM a podobně. Servery tak mají v síti výsadní postavení a jsou u nich kladeny vysoké nároky na výkon a spolehlivost. To vedlo firmu Novell k vývoji vysoce specializovaného operačního systému, který sice prakticky umí téměř jen sdílet své disky, ale zato to umí perfektně. Staví se tak do zcela vyhraněné pozice, kdy na rozdíl od Unixů nekoncentruje veškeré programy a prostředí pro jejich spouštění do jediného místa, ale přenechává maximální možnou míru povinností na stanicích. Ty musí mít vlastní operační systém a NetWare se stará pou-

ze o to, aby se stanicím zdálo, že všechna sdílená zařízení se nacházejí přímo u stanice samotné. Spuštěné programy pak běží přímo na jednotlivých stanicích a iluze se sdílenými zařízeními je pro ně tak dokonalá, že s nimi dokáží pracovat téměř všechny programy, i když pro provoz v síti původně vůbec nebyly napsány.

Před tím, než může uživatel z nějaké stanice využívat sdílených zařízení poskytovaných serverem, musí se k němu přihlásit, čili prokázat mu svou identitu pomocí jména a hesla. Pokud se uživatel úspěšně přihlásí, jsou mu na oplátku až do odhlášení poskytovány veškeré zdroje, které smí daný uživatel užívat. Práva k užívání jsou evidována na serveru a uživatelům je přiděluje správce (administrátor) systému.

Linux, se kterým pracujeme, má o mnoho větší možnosti, než server s Novell NetWare. Proto se nelze divit, že je v jeho silách dokonce emulovat činnost samotného NetWare serveru. Linux se pak navenek mimo jiné tváří, jako by byl NetWare serverem. Nedosahuje sice takových kvalit jako jeho specializovaný kolega, ale přesto ho může nahradit více než se ctí. Připočteme-li k tomu prakticky nulovou cenu a mnohostrannost samotného Linuxu, který můžeme současně používat na spoustu dalších věcí, začne být tento rys velice zajímavý.

Co je Mars a Linware?

Pokud se Linux chce stát zdánlivým NetWare serverem, musí umět komunikovat pomocí protokolu IPX, je třeba implementovat NCP protokol a musí se do sítě začít ohlašovat pomocí SAP protokolu. Čím lépe bude takto napodobovat svůj vzor, tím lépe pro nás. Tuto službu jsou schopny zastat dva programy: Mars a Linware.

Linware – český produkt

Autorem Linware (1) je Aleš Dryák z ČVUT v Praze. S vývojem skončil v roce 1995. V roce 1996 tento projekt ožil, když se Aleš sešel s Petrem Vandrovcem a Jirkou Novákem. J. F. Chadima pak provedl zásadní zásah do kódu, který vedl do slepé uličky a fakticky k zastavení projektu. Petr a Jirka pak ještě sice Linware trošku upravili, ale dnes na jeho dalším vývoji prakticky nikdo nepracuje. Linware je dnes ve funkčním stavu, ale pro jeho další pokračování je ho nutné prakticky celý přepsat. Zde se nabízí možnost pro kohokoliv, kdo by měl vůli tento zajímavý projekt a konkurenci Marsu vést dál.

Mars

Mars (2) (*Martin Stovers NetWare Emulator*) je původem z Německa a je neustále zdokonalován. Dnes Mars poskytuje téměř všechny služby jako Novell NetWare verze 3.12. Na jeho vývoji spolupracuje více lidí z celého světa a i Martin na projektu stále pracuje. Jak sám píše, hlavním důvodem, který ho přiměl k uvolnění Marsu do volného užívání, bylo uveřejnění Linware.

Společné rysy Marsu a Linware

Disky určené ke sdílení jsou ve skutečnosti určité adresáře v lokálním filesystému Linuxu a uživatelé, kteří se do emulátorů přihlašují, odpovídají každý normálnímu uživateli v Linuxu. NW přístupová práva pak vyplývají z Unixových práv uživatele k souborům a adresářům, protože jsou na uživatele uplatňována stejným způsobem, jako kdyby pracoval přímo v Linuxu (protože program, který soubory z disku čte a zasílá mu je na stanici, běží s jeho UID/GID). Ve sdílených adresářích lze použít symbolických linek, ale nelze je omezit na konkrétní podstrom.

Ani jeden emulátor nezavádí implementaci klasických NW práv k adresářům a souborům, se kterými jejich vzor výhradně pracuje. Vede to k jistým omezením, ale na druhou stranu to maximálně zjednodušuje funkci emulátorů. Nepříjemnost tak může způsobit program, který pátrá po právech v adresáři, protože díky této filozofii žádná práva nenajde i přes to, že má do adresáře přístup. Do budoucna se můžeme těšit na implementaci NW práv v Marsu, kterou již Martin Stower v konferenci oznámil.

Hesla pro Linux a pro emulátor jsou vedena zvlášť, protože při autentifikaci do emulátoru nevyhovuje systém používaný v Unixu, kde jsou hesla šifrována tak, že z uložené zakódované podoby nelze zpětně zjistit původní heslo. Výskyt dvou různých hesel na jednom počítači tak může být zdrojem drobných potíží pro uživatele systému. Na druhou stranu oba programy umožňují měnit heslo do emulátoru klasickými NW utilitami přímo ze stanice pomocí funkcí protokolu NCP stejným způsobem, jako měníme heslo na NetWare serveru.

Hlavní rozdíly mezi Marsem a Linware

Pokusím se stručně popsat hlavní rozdíly mezi oběma programy, které jsou vidět zvenčí. Nebudu se zabývat ani vlastní stavbou obou programů ani jejich testováním například na rychlost, stabilitu a jiná další měřítka. Spolu s vybudováním kvalitní metodiky je to námět nejméně na jednu větší samostatnou práci.

Mars je složen z více programů, ale navenek pracujeme jen s jedním. Každého připojeného uživatele obsluhuje samostatný proces. Má jen jeden konfigurační soubor, který je poměrně obsáhlý, přehledný a jednotlivé volby jsou v něm dobře dokumentovány. Nastavit lze téměř vše a jen velmi málo voleb je nastavitelných pouze při kompilaci, což přispívá k uživatelskému komfortu. Mars umí vytvořit tiskové fronty, do kterých můžeme zasílat tiskové úlohy a také tiskový server, který může odebírat úlohy z jiných NW tiskových front. Z front mohou být úlohy směrovány přímo do linuxového tiskového systému nebo lze využít jakýkoliv jiný tiskový server, který se dokáže na NW tiskové fronty napojit. Bindery databáze je uložena v samostatných souborech a obsahuje i zakódovaná hesla uživatelů. Mezi nejsympatičtější rys patří podpora dlouhých jmen, takže z Windows 95 lze pracovat i s dlouhými názvy (v NetWare je to tzv. OS/2 namespace) na NW discích. Dokumentace je přiměřená, v německém jazyce obsáhlejší.

Linware je jen jeden jediný démon, který sám obsluhuje všechny připojené uživatele. Potřebuje ovšem pro korektní běh dva další démony `IPXRIPD` a `IPXSAPD`. Konfigurační soubor je jeden a je oproti Marsu nepoměrně skromější. Linware nepodporuje tiskové fronty, nemá bindery databázi. Hesla jsou uložena v zakódované podobě ve speciálním souboru `/etc/lwpasswd`. Hlavní výhodou Linware je vlastnost, že umí zkracovat dlouhá jména a umožňuje tak i v prostředí DOSu editovat soubory s dlouhými jmény, což je výhodné například pro úpravy WWW stránek z Windows 3.x, které jsou na mnoha místech stále používány pro

svoji nenáročnost. Na druhé straně neposkytuje soubory s dlouhými názvy pomocí OS/2 namespace, což může být na překážku. Dokumentace je stručná. Linware potřebuje pro svou činnost patch na jádro, a tak se nevyhneme jeho kompilaci.

Bootování stanic pomocí BootROM

Abychom omezili šíření virů ve své počítačové síti, je vhodné i přes to, že máme v každé stanici harddisky, bootovat stanice pomocí BootROMek ze serveru. Obrazy bootovacích disket jsou na serveru chráněny před viry a my máme k dispozici po restartu stanice čistý a hlavně funkční počítač. BootROMky používají k otevírání obrazů na serveru stadardní funkce NCP protokolu, takže lze bootovat i z obrazů umístěných na emulátorech. Výjimkou jsou BootROMky, které používají výhradně RPL protokol. Ten není v Linuxu implementován a tak máme v tomto ohledu smůlu. Bootovat je samozřejmě možné i bezdiskové stanice.

Nasazení Marsu a Linware aneb jak je mám rád

Oba emulátory provozuji již poměrně dlouho. Dětské nemoci mají už za sebou a lze je poměrně seriózním způsobem využívat. Linware používám pro jeho schopnost zkracovat dlouhá jména souborů pro sdílení domácích adresářů Linuxu. Uživatelé tak mohou například snadno upravovat své WWW stránky z DOSu nebo Windows 3.x. Mars jsem nasadil do jedné učebny ve škole, kde je 11 bezdiskových stanic, které bootují pomocí BootROM a provozuje se na nich DOS a Windows 3.x. Linux nahradil NW server, běží na něm WWW server, proxy, obstarává dial-up a podobně. Subjektivním pozorováním lze říci, že ani po zdvojnásobení operační paměti z 16 na 32MB a výměně procesoru 486DX4/100 za Pentium 120 nedosahuje sdílení disků rychlosti původního NW serveru. Problémy jsou zejména v situaci, kdy stanice s Windows 3.x používají odkládací soubor na sdílených discích. Kontrolka HDD se rozsvítí a stanice na dlouhé vteřiny doslova ztuhnou. Problém bude nejspíše v optimalizaci přístupu k disku, která pro tento účel v Linuxu asi není nejvhodnější. Celkový přínos výměny OS (při dnešních cenách hardware) ovšem považuji přesto za kladný. Na ostatních místech, kde jsem Mars instaloval, pracuje více než uspokojivě, protože neplní úlohu jediného serveru se spoustou bezdiskových stanic, a nejsou na něj tak kladeny extrémní nároky.

Závěr

Pokud si myslíte, že Mars nebo Linware může plně nahradit NetWare servery od Novellu, zažijete nejspíše zklamání. Profesionální řešení od firmy Novell vyniká v mnoha ohledech, zejména však v rychlosti, nenáročnosti na hardware a komplexnosi nabízených služeb. Konkurovat však nemůže v požadavcích na flexibilitu, univerzálnost a hlavně cenu. To je místo, kam Mars a Linware patří a kde jistě cítíte i vy sílu Linuxu. Nezbývá, než se smířit s daným stavem věcí a využít především silné stránky každého řešení. ■

1 Linware
ftp://platan.vc.cvut.cz/pub/linux/lwared-0.95/
2 Mars
http://www.compu-art.de/download/mars_nwe.html

Co nového na sunsite.unc.edu?

Pavel Janík ml., 30. dubna 1998

X11

X11/xutils/modeselect-1.0.tar.gz – menu pro přihlašovací obrazovku XDM umožňující zvolit hloubku displaye

apps

apps/doctools/man/man-1.5d.tar.gz – nová verze prohlížeče manuálových stránek podporující kompresi

apps/editors/X/code_crusader-0.13.0_shared.tar.gz – vývojové prostředí pro C/C++

apps/editors/X/fte-0.46b4.src.zip – editor umožňující zvýrazňování syntaxe a inteligentní odsazování

apps/editors/X/nedit-5.0.2-source.tar.gz – programátorský editor

apps/editors/X/xwpe-1.5.8a.tgz – známý klon Borland IDE

apps/editors/X/yudit-0.99.tar.gz – textový editor pro X-Windows podporující unicode

apps/graphics/viewers/X/Upm_linu.tgz – prohlížeč souborů ve formátu HPGL

apps/tex/simple-2.1.2.tgz – preprocessor upravený pro TEX

apps/tex/ts-9804.tgz – TeX shell pro X-Window System

apps/tex/xgod-1.1.tar.gz – Go diagramy pro TEX

apps/www/servers/shellver-0.0.2.tar.gz – malý a jednoduchý WWW server

commercial

commercial/dbox-1.60.tgz – BBS balík pro Linux

commercial/nftp-1.21-linux-x86.tar.gz – klient FTP se záložkami

devel

devel/lang/c/cxref-1.4b.tgz – křížové odkazy v jazyce C

docs

docs/linux-doc-project/man-pages/man-pages-1.19.tar.gz – nové manuálové stránky pro Linux

games

games/amusements/divination/xbio-2.0.tar.gz – biorytmy

games/strategy/phalanx-14.tar.gz – nová verze šachového programu českého autora

hardware

hardware/pciutils-1.03.tar.gz – PCI utilitky Martina Mareše

science

science/visualization/plotting/xFgraphs-1.0.tar.gz – grafy funkcí pro X-Window System

```
Summary: graphics memory usage meter
Name: gmemusage
Version: 0.2
Release: 1
Source: http://reality.sgi.com/raju/software/gmemusage-0.2.tar.gz
Copyright: GPL
Group: Utilities/System
Packager: Jan "Yenya" Kasprzak <kas@fi.muni.cz>
BuildRoot: /tmp/gmemusage-root
%description
This tool displays the bar graph describing memory usage of processes
on the Linux box. Uses /proc filesystem.
%prep
%setup
%build
make OPTIM="$RPM_OPT_FLAGS"
%install
mkdir -p $RPM_BUILD_ROOT/usr/X11R6/{bin,man/man1}
make PREFIX=$RPM_BUILD_ROOT/usr/X11R6 install
%files
%attr(0775,root,root) /usr/X11R6/bin/gmemusage
%attr(0644,root,root) /usr/X11R6/man/man1/gmemusage.1
```

Výpis č. 6: Soubor SPEC pro gmemusage

system

system/backup/tbackup-0.9.tgz – zálohovací program s možností inkrementálního zálohování

system/bbs/forums-2.0.tgz.tgz – BBS v Perlu

system/network/admin/firewallct-1.0.8.tar.gz – konfigurace firewallu přes HTML rozhraní

system/printing/powercolor-0.3.tar.gz – GUI pro konfiguraci různých parametrů tiskáren HP Deskjet

utils

utils/compress/unrar-2.03.1.tar.gz – RAR pro Linux

■

Tvorba RPM balíků

Jan Kasprzak, 15. května 1998

V předchozích dílech našeho seriálu o systému RPM jsme rozebírali všechny vlastnosti RPM z hlediska uživatele. V další části se budeme věnovat RPM z hlediska vlastní tvorby RPM balíků. Co tedy dělat, máme-li nějaký software a chceme jej zabalit do RPM balíku?

Kompilace

První činnost, kterou je nutno před vlastním zabalením softwaru do formátu RPM udělat, je zkompilovat software pod Linuxem a ujistit se, jestli vůbec pod Linuxem funguje. V následujícím budeme předpokládat, že software lze bez problémů zkompilovat a že také funguje. Jako příklad si vezmeme program `gmemusage`, který slouží k měření obsazení operační paměti jednotlivými procesy. K jeho zkompilování a nainstalování stačí spustit příkazy `make` a `make install`.

Rychlý start

Bez jakéhokoli vysvětlování uvedu příklad, jak vyrobit RPM-balík `gmemusage` (vysvětlení bude následovat):

- Jako root nakopírujte soubor (1) do adresáře `/usr/src/redhat/SOURCES/`.
- Nakopírujte soubor na výpise Soubor SPEC pro gmemusage pod názvem `gmemusage-0.2.spec` do adresáře `/usr/src/redhat/SPECS/`.
- V adresáři `/usr/src/redhat/SPECS` spusťte příkaz `rpm -ba gmemusage-0.2.spec`. Výsledkem by měl být zdrojový RPM soubor `gmemusage-0.2-1.src.rpm` v adresáři `/usr/src/redhat/SRPMS` (k čemu jsou zdrojové RPM soubory, uvidíme později) a binární RPM pro danou architekturu (`gmemusage` jsem zkoušel vyrábět pouze pro platformu `i386` a `sparc`, ale pro `sparc` měl program jisté problémy. Binární RPM soubor je v `RPMS/i386/gmemusage-0.2-1.i386.rpm`.
- Nově vytvořený RPM soubor lze nainstalovat příkazem `rpm -Uvh RPMS/i386/gmemusage-0.2-1.i386.rpm`.

Vysvětlení

A teď zpět k tomu, co jsme vlastně v předchozí sekci udělali. Je vidět několik základních faktů:

- RPM využívá adresář `/usr/src/redhat`.
- používá se příkaz `rpm -ba`

- tvorba balíku je řízena souborem s koncovkou `.spec`.
- vytváří se zároveň zdrojový i binární RPM soubor.

Adresář `/usr/src/redhat` obsahuje tyto podadresáře (lze změnit v souboru `/etc/rpmrc`, což umožní vyrábět RPM balíky i jako běžný uživatel):

- SPECS – obsahuje spec-soubory, které řídí vlastní sestavování RPM balíku.
- SOURCES – zde jsou zdrojové soubory, ze kterých se balík vytváří.
- BUILD – do tohoto adresáře se rozbalí zdrojové soubory a probíhá zde kompilace.
- RPMS/architektura – sem se uloží výsledné RPM soubory.
- SRPMS – zde budou výsledné zdrojové RPM soubory.

SPEC soubor

Tento soubor řídí celou kompilaci. Jeho příklad byl uveden, jednotlivé možnosti budou podrobně popsány v následujících dílech tohoto seriálu.

Na začátku souboru je hlavička, která popisuje, o jaký balík se jedná, odkud pochází zdrojový text, kdo balík vytvořil, jaká je jeho licence, a množství dalších informací, později z balíku vypsatelných pomocí `rpm -q`.

Dále následují jednotlivé sekce — jsou odděleny klíčovým slovem s procentem na začátku řádku. V našem příkladu jsou použity tyto sekce:

- `%description` – popis balíku, který dále rozvíjí krátkou informaci ze `Summary:` v hlavičce souboru.
- `%prep` – příprava na kompilaci. Tato sekce se vykoná jako shellovský skript a má za úkol provést rozbalení zdrojového archívu, aplikaci případných záplat, konfiguraci a podobně. Ve výše uvedeném souboru tato sekce obsahuje jediný příkaz – makro (pozor, ne sekci) `%setup`, které dělá přesně to, že do adresáře BUILD rozbalí archív, uvedený v hlavičce jako `Source:` z adresáře SOURCES. Tedy z celého tam uvedeného URL se použije pouze část za posledním lomítkem. Ostatní slouží jen pro informaci.
- `%build` – vlastní kompilace. Jde opět o shellovský skript. Zde je dobré si všimnout proměnné `$RPM_OPT_FLAGS` – jednotlivé skripty dostávají od RPM přednastavené některé proměnné. Tato proměnná je jednou z nich a obsahuje implicitní volby předávané kompilátoru pro optimalizaci. Takže změnou tohoto parametru například v `/etc/rpmrc` můžeme dosáhnout toho, aby se všechny nově kompilované balíky stavěly s těmito volbami.
- `%install` – vlastní instalace souborů v balíku. V běžných případech se v tomto skriptu spouští příkaz `make install`. Je ovšem dobré, pokud rekompilace RPM balíku neprovede instalaci do skutečného systému, ale někam do vlastního stromu. Tato vlastnost se jmenuje Buildroot a zapíná se uvedením stejnojmenného parametru v hlavičce spec-souboru. RPM pak vytvoří jmenovaný adresář a nastaví skriptům (včetně toho `%install`) proměnnou `$RPM_BUILD_ROOT`. Výhoda tohoto přístupu je v tom, že takto může bez modifikace spec-souboru vytvořit RPM balík i běžný uživatel, nikoliv jen superuživatel.
- `%files` – popisuje, které soubory mají být zahrnuty do RPM balíku. Tato sekce není shellovský skript, ale seznam. Je možno též specifikovat atributy jednotlivých souborů, čímž umožníme vytvořit RPM balík i běžnému uživateli. Neuvedeme-li atributy, RPM vezme takové atributy, jaké má jmenovaný soubor v době tvorby balíku.

Příkaz rpm –b

K vytváření RPM balíků se používá příkaz `rpm -b`. Jeho argumentem je `.spec` soubor. Akceptuje následující přepínače:

- `-p` Vykoná pouze `%prep` sekci ve spec-souboru.
- `-l` Zkontroluje, jestli soubory jmenované v sekci `%files` skutečně existují.
- `-c` Vykoná `%prep` a `%build`.
- `-i` Vykoná `%prep`, `%build` a `%install`.
- `-b` Vytvoří binární RPM soubor (po vykonání `%prep`, `%build` a `%install`).
- `-a` Vytvoří binární a zdrojový RPM soubor (po vykonání `%prep`, `%build` a `%install`).
- `--short-circuit` Používá se ve spojitosti s `-bc` a `-bi`. Vykoná pouze jmenovanou sekci, bez předchozích.
- `--timecheck číslo` Vypíše varování, pokud se snažíme do RPM balíku zabalit soubor starší než daný počet sekund.
- `--clean` Smaže kompilační strom příslušného balíku v adresáři `/usr/src/redhat/BUILD` po skončení tvorby balíku.
- `--test` Otestuje syntaxi spec-souboru, neprovádí žádné kompilační akce.
- `--sign` Vytvoří balík s PGP podpisem.

Kromě `rpm -b` lze také použít `rpm -t` a jako parametr uvést soubor ve formátu `tar.gz`. Takovýto archív se rozbalí a jako spec-soubor se použije první soubor s koncovkou `.spec`, obsažený v archívu. Při kompilaci se samozřejmě neprovádí sekce `%prep`.

Zdrojové RPM soubory

Již několikrát padla zmínka o zdrojových RPM souborech. To je soubor s koncovkou `.src.rpm`. Takovýto soubor obsahuje v sobě spec-soubor příslušného balíku, dále všechny soubory potřebné k sestavení binárního RPM balíku (zejména soubory uvedené v hlavičce v položce `Source`) a případnou ikonku RPM balíku.

Zdrojový RPM balík lze zpracovávat třemi způsoby. První možnost je tento balík nainstalovat pomocí `rpm -i`. Všechny soubory, které zdrojový balík obsahuje, jsou nainstalovány do adresáře SOURCES a spec-soubor je poté přesunut do adresáře SPECS. Další dvě možnosti jsou vytvoření binárního RPM balíku pro danou architekturu, případně vytvoření jak binárního, tak

i zdrojového RPM balíku. Dosáhneme toho příkazem `rpm --recompile balík.src.rpm`, případně `rpm --rebuild balík.src.rpm`. Oproti příkazu `rpm -ba` se navíc implicitně smaže kompilační adresář tak, jako kdybychom použili `--clean`.

To zatím jako úvod do problematiky tvorby vlastních RPM balíků stačí. Příště se podrobněji vrátíme k formátu spec-souboru a k podrobnostem této problematiky. ■

1 Zdrojový text programu gmemusage
http://reality.sgi.com/raju/software/gmemusage-0.2.tar.gz

LINUX — Internet server (druhé upravené vydání)

Michal Polák, 22. dubna 1998

Po čtenářsky velmi úspěšném vydání knihy LINUX — Internet server přichází na pulty druhé upravené vydání. Tato knížka dostala od redakce českého CHIPu ocenění „CHIP tip“. Hned v úvodu musím podotknout, že mi tato kniha poskytla mnoho poučení a tedy tento článek bude laděn v opačném duchu, než bývá většina recenzí Mirky Spáčilové v MF DNES…

Celá publikace je napsána čtivou formou a místy okořeněna vtipnými poznámkami. Pro znalce prvního vydání pouze uvedu, že v druhém vydání chybí podkapitola o Gopheru (používá to ještě dneska někdo?) a naopak přibyla zmínka o beztřídních adresách a podkapitola s názvem „Za Web interaktivní“, ostatní zůstává v původním znění. Všem, kteří se s touto knihou ještě nesetkali a pracují s Linuxem, vřele doporučuji nenechat si tuto knihu ujít a mít ji v pohotovosti, nejlépe v blízkosti počítače. V počátku se vám ani nestihne zaprášit. :-) Nejenom administrátoři by měli vědět, jak zhruba zevnitř funguje Internet a sítě vůbec. Pokud platí vztah uživatel LINUXu = administrátor (aspoň na lokálním stroji), je minulá věta bezpředmětná.

Kniha je rozdělena do tří částí: „Transportní systém“, „Aplikační služby“ a „Sloužím lidu“.

Kapitola „Transportní systém“ je technologicky zaměřena, tj. zabývá se přenosovými protokoly, konfigurací služeb a dalšími mechanismy, které jsou nezbytné k provozování služeb Internetu. Čtenář se seznámí se síťovými vrstvami, problémem IP adres — třídy, přidělování, podsítě, masky podsítí, beztřídní adresy, směrování, základní mechanismy TCP.

Přičemž pojmem beztřídní adresa se vracíme zpět ke kořenům, kdy neexistovalo rozdělení IP adres na třídy A, B, C, D. Rychle rostoucí počet zapojených sítí a počítačů si vynutil nový přístup k adresám, kdy adresa je rozdělena pouze na dvě části — adresa sítě a lokální adresa v ní.

Další část kapitoly se zabývá zprovozněním TCP/IP (`ifconfig`, `route`, `ping`, `traceroute`, `gated`), připojením sériovou linkou SLIP, PPP, podrobným popisem DNS, konfigurací portů, synchronizací času, nezapomíná se ani na bezpečnost — TCP wrapper, firewall.

Na závěr kapitoly se dostala správa sítě (ve smyslu kabeláže, zapojování a konfigurace počítačů, aktivních síťových prvků a dalších zařízení, kontrola jejich činnosti), kterou ocení zejména příznivci okenního ovládání. Pojednává se totiž o programech `tkined` a `scotty`, ve kterých si mj. můžete schéma vaší sítě vlastnoručně namalovat. Vše je realizováno v protokolu SNMP, je tedy zapotřebí mít ještě SNMP agenty.

Kapitola „Aplikační služby“ se zabývá zprovozněním jednotlivých informačních služeb. V první řadě se jedná o `telnet` a `rlogin`. Kniha však zcela oprávněně vyzdvihuje a doporučuje používat raději *Secure Shell*. Dalším tématem je elektronická pošta — od popisu předávání zpráv, struktury zpráv, konfigurace, až po diskuse o `sendmailu`, IDA rozšíření a aliasech. Následuje pojednání o UUCP a elektronických skupinách — zde jsou zmíněny programy `listproc`, `Majordomo`, TULP. My se ale zastavíme až u popisu služby FTP, která je zde zdůrazněna coby nejmenší společný jmenovatel pro zveřejnění statických informací na Internetu. Zde se čtenář dozví nejenom mechanismus fungování, ale i detailní konfiguraci a několik užitečných rad — např. automatickou konverzi souborů (pomocí `wu-ftpd`). Z doplňkových služeb k FTP je ještě zmíněn program Index Master (binárka se jmenuje `index`) na automatickou tvorbu informací o souborech v jednotlivých adresářích, či programy `mirror` a `mirror-master`. K čemu asi slouží?

> Díky okénkům (sliding windows) může odesílatel vyslat několik TCP segmentů a nečekat před každým, až dorazí potvrzení předchozího. Výsledkem je významné zrychlení komunikace, ovšem s lidskou tváří — příjemce si diktuje, kolik dat snese. Sociální demokracii by se takový přístup jistě zamlouval.

Nejvýznamnější a nejrozsáhlejší podkapitolou je samozřejmě WWW. Po úvodním zahřívacím seznámení se se základními pojmy přichází na řadu popis serveru Apache, který prý byl na začátku roku 1998 nasazen na 45% všech WWW serverů na světě. V době psaní textu knihy již byl k mání Apache 1.3 beta, který by měl podporovat i Microsoft Windows 95 a NT, ale autoři dali raději přednost popisu Apache 1.2.5. Rozebírá se zde vše od vlastní konfigurace, problematiky CGI scriptů, přes omezování uživatelů či dosahu příkazů až po skutečné lahůdky, jako je např. virtuální server, vkládání vsuvek serverem nebo podmíněné vkládání částí dokumentů nebo konfiguraci PROXY serveru. Problém češtiny na Webu je rozepsán do samostatné podkapitoly.

Nově přibyvší podkapitola „Za Web interaktivní“ obsahuje zmínku o jazyku PHP — pro stejně neznalé jako já — *Proffessional Home Page*. Jedná se o jednoduchý jazyk, který se zapisuje jako součást WWW stránky, ale tyto instrukce se ke klientovi nedostanou, server je sám vykoná. Na jejich základě vloží do stránky dílčí informace, obsah jiného souboru či výsledek databázového dotazu. Možnost pracovat s SQL dotazy slibuje prudký rozmach PHP.

Poslední podkapitolou jsou informace o provozu a konfiguraci „Usenet News“.

Třetí kapitola „Sloužím lidu“ popisuje služby Linuxu pro počítače uživatelů v lokální síti. Hlavně seznamuje s emulátory známých nástrojů pro sdílení prostředků. Jde hlavně o NetWare, Sambu (pro Microsoft Windows) a AppleTalk.

Na závěr jsem pro vás vybral ještě jeden z výroků aspirujících na zařazení do rubriky „Zasmáli jsme se“.

> Pokud jste nikdy neupravovali sendmail.cf, nejste skutečný správce Unixu.
> Jestliže jste to udělali víc než jednou, jste blázen.

Věřím, že se vám bude kniha líbit. Pokud si ji chcete zakoupit, samozřejmě můžete. Měla by být k dostání v kaž-

dém počítačovém knihkupectví a také si ji můžete objednat přímo na serveru společnosti Neokortex (1). ■

1 Neokortex, spol. s r. o.
http://www.neo.cz

Zasmáli jsme se!

Pavel Janík ml., 12. května 1998

Smích je kořením života, a proto hned začneme jedním vtipem:

> *Otázka*: Tak už se nám po Linuxu dostaly do kosmu i Windowsy. Přímo na měsíc. A víte, co tam dělají?
> *Odpověď*: Padají 6x pomaleji…
> Jan Staníček v konferenci *jokes@satoya.cz*

Být přihlášen v konferenci *jokes@satoya.cz* se opravdu vyplatí ;-)

> ... to byl čí nápad, že se Windows 95 budou končit stisknutím tlačítka Start?

Ale ani v konferenci *linux-kernel@vger.rutgers.edu* není nouze o humor. Tento měsíc jsem se však smál pouze jednou, protože došlo k poškození seznamu přihlášených, což jsem postřehl bohužel až po dvou dnech, kdy jsem měl v přihrádce `LinuxKernel` pouhých 5 dopisů. Ale to jen tak na okraj. Tak tedy, kdy jsem se zasmál? Začneme od začátku. Alan Cox ve svých dopisech nikdy nepoužívá signaturu. Tento měsíc ji jednou použil…

> Knowledge is power
> Information is a weapon
> Truth is an inconvenience
> Welcome to the corporate mindset
> Free Software, Free Speech
> Free Information, Free Association
> Free Thought, Free World
> Free Your Mind
>
> ---
>
> Vědění je síla
> Informace je zbraň
> Pravda je nepříjemnost
> Vítejte v komerčním myšlení!
> Free software, svoboda projevu
> svoboda informací, svoboda shromažďování
> svobodné myšlení, svobodný svět
> Osvoboďte svou mysl!
> Signatura Alana Coxe

Milana Keršlágera už nebaví (stejně jako i některé další) číst v *linux@muni.cz* příspěvky, které je možno zodpovědět po tříminutovém hledání na některém z vyhledávacích serverů. Milan k tomu dnes poznamenal:

> Oč by svět byl jednodušší, kdyby všichni uměli číst! ;-)

■

Linuxové noviny a jejich šíření

Linuxové noviny vydává České sdružení uživatelů operačního systému Linux pro své příznivce a sympatizanty.

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 (1). 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.

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

Linuxové noviny můžete najít na akademické síti CESNET (2), na síti IBM Global Network na adrese (3), na serveru časopisu Netáčik (4), který je připojen do slovenského SIXu, případně na serveru společnosti OptiCom (5).

Linuxové noviny jsou k dispozici také ve formátu HTML na adrese (6). ■

1 Adresa redakce
mailto:noviny@linux.cz
2 Linuxové noviny na síti CESNET
ftp://ftp.fi.muni.cz/pub/linux/local/noviny
3 Linuxové noviny na síti IBM Global Network
ftp://ftp.inet.cz/pub/People/Pavel.Janik/noviny
4 Slovenské zrcadlo Linuxových novin
ftp://netacik.sk/pub/linux/cz-noviny
5 Linuxové noviny – OptiCom
http://www.mathew.sk/noviny
6 Linuxové noviny ve formátu HTML
http://www.linux.cz/noviny