Abíčko

Časopis serveru AbcLinuxu.cz

září 2002

Obsah

Editoriál

Vítejte u čtení časopisu Abíčko.

Abíčko vychází jako měsíční příloha serveru *AbcLinuxu.cz* a obsahuje výběr toho nejzajímavějšího obsahu, který zde byl v minulém měsíci publikován. Touto formou chceme předat čtenářům informace v snadno čitelné podobě vhodné i pro tisk.

Cílem serveru AbcLinuxu.cz (založen na jaře 1999 pod názvem Linux Hardware) je pomáhat všem uživatelům Linuxu, nezávisle na jejich zkušenostech, platformě či použité distribuci. Motorem, který nás pohání vpřed, je idea vzájemné pomoci a spolupráce. Proto i velkou část obsahu tvoří samotní uživatelé. Zapojit se může kdokoliv, tedy i vy.

Na AbcLinuxu.cz najdete rozsáhlou databázi hardwaru (návodů na instalaci pod Linuxem), velice aktivní diskusní fórum, podrobné návody a tutoriály, recenze, archiv ovladačů, informace o linuxovém jádře (včetně populárních Jaderných novin) i rozcestník po ostatních linuxových serverech. Další služby jsou v přípravě (včetně služeb pro firmy).

Máte-li námět na článek, zašlete jej do konference našich autorů: *autori@abclinuxu.cz*. Máte-li zájem o sponzoring Abíčka nebo jinou formu reklamy, kontaktujte nás na adrese *reklama@abclinuxu.cz*. Ostatní dotazy směrujte na adresu *literakl@abclinuxu.cz*.

Sazba: Ondřej Krejčík

Tučňák má ostré modré zuby!

CIJOML

Úvod

Doufám, že jsem vás názvem článku nezastrašil. Nebojte se. Naši tučnáci nekoušou. Právě naopak. Dalším oborem, kde nám tučnáci dělají radost, je technologie Bluetooth (modrý zub). Dnes si povíme něco o tom, jak našeho tučnáčka vybavit pro cestování s notebookem. Samozřejmě se dá použít jakýkoliv počítač s USB, ale bezdrátová komunikace dostává tu správnou šťávu až při využití na cestách.

Je přeci úžasné si jen tak položit v autě na klín notebook, telefon mít v kapse a nemuset řešit kabel, do kterého se v lepším případě zamotáte a v horším se na něm oběsíte, neřkuli infraport, který musíte neustále přesvědčovat, že opravdu není dobrý nápad přestat se koukat vašemu notebooku do červeného očička právě v momentě, kdy šéfovi odesíláte nejdůležitější e-mail za půl roku. O špatné konstelaci hvězd, kdy si způsobíte vypálení v lepším případě jednoho a v horším případě obou očí, ani nemluvě :-).

Asi se divíte, proč zrovna modrý zub. Nejprve si povíme něco o vzniku názvu této technologie, protože ten je stejně zajímavý, ne-li zajímavější, než technologie celá. Název Bluetooth vznikl dle krále Haralda, zvaném Modrozub (Bluetooth), který na přelomu 10. století sjednotil Dánsko a Norsko. Jeho velikou předností byla schopnost komunikace s lidmi. Povedlo se mu do té doby něco nevídaného. Připojil ke své říši území bez války jen na základě dohody. Nejspíš proto si společnost Ericsson, která tuto technologii světu dala a která stojí v čele sdružení prosazující tento standard, vzpomněla na tohoto velikána téměř svých dějin (sídlo společnosti je o pár set kilometrů severovýchodně a to ve Švédsku).

Vlastnosti

Proč tato technologie vznikla? Odpověd je nasnadě už z prvního odstavce. Prostě tato technologie se snaží o tyto věci:

- eliminaci kabelů a drátů nutných pro propojení s nehybnými i mobilními zařízeními
- podporu datové i hlasové komunikace
- využívání možností Ad hoc sítí za předpokladu nejvyšší synchronizace mezi komunikujícími zařízeními

A jak takové bluetooth zařízení vypadá? Jedná se o malý čip, velikosti 9x9 milimetrů, komunikující v rádiovém pásmu 2.4 GHz. Následují technické detaily čipu Bluetooth:

- Zakladatel: Ericsson
- Cena čipu: 20 dolarů
- Rozměry čipu: 9 x 9 milimetrů
- Spotřeba: stand-by 0,3 mA; při přenosu 30mA
- Provozní pásmo: 2402 až 2483.5 MHz
- Přenosová rychlost: 1 Mb/s (možnost komunikace synchronní a asynchronní)
  - asynchronní: uplink 721 kb/s, downlink 57,6 kb/s
  - synchronní: 432,6 kb/s oběma směry
  - uvedené rychlosti jsou čisté, tj. po započítání řídících a opravných protokolů
- Dosah: až 100 metrů (se zesilovacím stupněm) - bez něj 10 metrů

- Identifikace: 48-bitová adresa z IEEE 802.11 standardu
- Zabezpečení: 128bitovým šifrováním
- Frequency hopping: 1600 skoků/s
- Vzdálenost kanálů: 1 MHz ⇒ 79 kanálů
- Možné stavy zařízení: master, slave
- Počet dalších najednou komunikujících zařízení přes jeden čip: 7
- Možnost tvořit pikosítě, kdy 1 zařízení ve stavu slave je připojeno ke 2 masterům
- Spolupracující firmy: Ericsson, 3Com, Casion, Intel, Toshiba, Nokia, Motorola a stovky dalších

Kolik je na světě zařízení s podporou Bluetooth? V době psaní tohoto článku bylo dostupných 622 zařízení (*http://qualweb.opengroup.org/Template2.cfm?LinkQualified=QualifiedProducts*).

A nyní se podíváme na specifikaci zařízení. Každé zařízení podporuje takzvané profily. To je seznam funkcí, které zařízení zná a skrze které komunikuje se svým okolím. Profilů je několik desítek, ale každé zařízení zná jen několik. Proto je nutné si nejprve na výše uvedené stránce zkontrolovat, zdali vaše zařízení spolu budou komunikovat. Jinak se maximálně najdou, ale poté již nebude další komunikace možná.

Já zvolil tyto 2 zařízení: mobilní telefon Nokii 6310 a USB adapter Mitsumi typu WIF-0402C. Jak si můžete ověřit na zmíněné stránce, shodují se hned v několika profilech, takže komunikace bude velice snadná. A také byla.

Instalace

První, co následovalo, bylo vložení USB adaptéru do portu a návštěva značkového servisu Nokia, kde jsem si nechal nahrát poslední verzi firmware 4.20. Poté jsem věděl, že mušky ze strany telefonu budou minimalizované a já budu moci nadávat pouze sám sobě, když něco nebude fungovat.

Současné distribuce nemají přímo v sobě moduly pro bluetooth komunikaci, proto následovalo přeložení kernelu poslední verze 2.4.19-pre10 s podporou Bluetooth:

```
Bluetooth subsystem support
L2CAP protocol support
SCO links supportSCO links support
HCI USB driver
[*] Firmware download support
```

Dále jsem se snažil najít v distribuci Debian potřebné ovládací utility. Ty tam bohužel také nebyly, a tak jsem navštívil stránku *http://bluez.sf.net*, kde je projekt hostovaný a stáhnul tyto soubory:

- bluefw-0.3.tar.gz
- bluez-libs-2.0-pre9.tar.gz
- bluez-sdp-0.4.tar.gz
- bluez-utils-2.0-pre9.tar.gz
- hcidump-1.2.tar.gz
- rfcommd-1.1.tar.gz

Všechny soubory jsem rozbalil a jal se překládat. Jako první je potřeba přeložit bluez-libs. Pod distribucí debian nebyla kompilace možná. Probíhá ale v pořádku pod distribucí RedHat, na které je také primárně projekt vyvíjen. Proto jsem ze stránek *bluez.sf.net* stáhnul deb balíček a nainstaloval. Kompilace všech ostatních aplikací již poté byla bez problému. Je samozřejmě nutné stáhnout jak knihovny, tak i jejich devel verze.

Nyní je už jen třeba vše přinutit fungovat… Začneme inicializací adaptéru:

```
modprobe hci_usb; modprobe l2cap
hciconfig hci0 up
```

Zapneme podporu Bluetooth na Nokii (menu 10) a necháme naším zařízením nokii nalézt:

```
hcitool -i hci0 inq
```

Dostaneme tuto odpověď:

```
Inquiring ...
00:02:EE:00:AB:B3 clock offset: 0x54b6 class: 0x502204
```

Telefon byl tedy nalezen. Zkusíme ho pingnout, zda-li funguje:

```
l2ping 00:02:EE:00:AB:B3
Ping: 00:02:EE:00:AB:B3 from 00:A0:96:1F:B0:0C (data size 20) ...
0 bytes from 00:02:EE:00:AB:B3 id 200 time 29.46ms
0 bytes from 00:02:EE:00:AB:B3 id 201 time 26.83ms
2 sent, 2 received, 0% loss
```

I zde je tedy vše v pořádku a přistoupíme k nastavování komunikace.

Konfigurace

Jako první musíme nastavit **hcid**, což je démon, starající se o výměnu hesel mezi oběma zařízeními a jejich spárování.

```
cijoml@notas:~/bluetooth$ cat /etc/bluetooth/hcid.conf
#
# HCI daemon configuration file.
#
# $Id: hcid.conf,v 1.1.1.1 2002/03/08 21:12:35 maxk Exp $
#

# HCId options
options {
  # Automaticaly initialize new devices
  autoinit yes;
```

```
  # Security Manager mode
  # none - Security manager disabled
  # auto - Use local PIN for incomming connections
  # user - Always ask user for a PIN

  security auto;

  # PIN helper
  pin_helper /bin/bluepin;
}

# Default settings for HCI devices
device {
  # Local device name
  # %d - device id
  # %h - host name
  # name "BlueZ (%d)";
  name "Linux";

  # Local device class
  class 0x100;

  # Default packet type
  pkt_type DH1,DM1,HV1;

  # Inquiry and Page scann
  iscan enable; pscan enable;

  # Default link mode
  # none - no specific policy
  # accept - always accept incomming connections
  # master - become master on incomming connections,
  # deny role switch on outgoint connections
  #
  #lm accept,master;
  lm master;

  # Default link policy
  # none - no specific policy
  # rswitch - allow role switch
```

```
  # hold - allow hold mode
  # sniff - allow sniff mode
  # park - allow park mode

  #lp hold,sniff;
  lp hold,sniff,park;

  # Authentication and Encryption
  #auth enable;
  #encrypt enable;
}
```

Nyní je nutné nastavit démona, který pro nás obstarává komunikaci přes bluetooth – **rfcommd**. Démon bude nastaven jako slave, master bude telefon (pro snadnější nastavování).

```
cijoml@notas:~/bluetooth$ cat /etc/bluetooth/rfcommd.conf
options {
  psm 3;
  ppp /usr/sbin/pppd;
  ifconfig /sbin/ifconfig;
  route /sbin/route;
  firewall /sbin/iptables;
}

na {
  channel 1;
  up {
    ppp "%d call gprsbt";
  }
}
```

Nastavíme komunikaci přes ppp protokol. Zde bych upozornil, že není třeba cokoliv měnit ve vašich souborech `provider` a `chatscripts`, jen vymažte zmínku o `/dev/tty` zařízení. To pro vás bude simulovat **rfcommd**. Vše ostatní ponechte včetně rychlosti portu 115200.

Nyní je vše nastavené a komunikace, pakliže máme funkční ppp protokol, může začít:

```
rfcommd -n -f /usr/local/etc/rfcommd.conf na 00:02:EE:00:AB:B3
```

Nokia oznámí připojené zařízení Linux, zvolíme OK, a zadáme číslo. Toto si nemusíme pamatovat, zadává se pouze jednou.

Abychom nemuseli řešit problém, že na náš desktop se nám připojuje aplikace běžící pod rootem (hcid), zadáme **xhost** +. Vyjede na nás okénko, kde zadáme to samé číslo a potvrdíme. Zařízení jsou spárována, což nám Nokia oznámí a začne vytáčet a následně se spojí. Je dobré na Nokii vypnout u spárovaného zařízení požadavek autorizace spojení, abychom nemuseli toto potvrzovat při každém připojení přes notebook.

Od té doby má náš tučňák opravdu ostré modré zuby :-). Užijte si je, je to radost. A bude ještě větší, až se i u nás začnou prodávat tiskárny nebo chladničky s podporou této technologie.

Diskuse

Pepe: Howto již existuje: *http://bluez.sourceforge.net/howto/index.html*.

Jimak: Koho by to zajímalo, BlueZ není jediný a ani zdaleka nejlepší Blutooth stack pro Linux. Kdo se koukne na stránky konkurenčního Affixu (*http://affix.sourceforge.net*), tak zjistí, že se s tím dá dělat ještě mnohem a mnohem více. Instalace je totálně bezproblémová (vyvíjeno paralelně na Red Hatu a Debianu), k dostání jako jeden tar.gz (deb a rpm balíčky jsou taky), funguje se všemi kernely 2.4.x, není třeba nic patchovat.

CVS na dálku

Leoš Literák

Úvod

Velká část Open Source projektů využívá jako software na správu verzí CVS. Existují sice lepší alternativy, ale CVS je v této oblasti téměř standardem. Nainstalovat CVS není velký problém, v době RPM či PKG to zvládne každý. O něco složitější je situace, kdy chcete k CVS přistupovat vzdáleně. A touto tématikou se zabývá tento článek.

Existuje více způsobů, jak přistupovat vzdáleně k CVS. Jsou to přístup přes rsh, autentifikace přes heslo (pserver), CSSAPI a Kerberos. Rsh má smysl leda na uzavřené síti (třeba za firewallem), kde si všichni uživatelé navzájem věří. O něco lepším řešením je autentifikace uživatelů přes Kerberos, CVS zvládá jeho verzi 4. S nástupem verze 5 autoři vytvořili obecné rozhraní CSSAPI. Pro mé účely se však nejvíce hodí pserver. Je snadný na instalaci a konfiguraci a zároveň umožňuje vytvořit si virtuální uživatele či vymyslet existujícím uživatelům hesla platná jen pro CVS. Takže pokud útočník bude odposlouchávat vaši relaci, s CVS heslem váš server nenaboural.

Systémová část

První část konfigurace pserveru musíte provést jako root. Je třeba přidat do síťových služeb CVS. Do souboru `/etc/services` vložte tuto řádku:

```
cvspserver 2401/tcp
```

a pokud používáte démon **inetd**, pak do souboru `/etc/inetd.conf` tuto:

```
cvspserver stream tcp nowait root /usr/bin/cvs cvs -f --allow-root=/home/literakl/
/CVSROOT pserver
```

Moderní distribuce preferují démon **xinetd**, takže budeme muset vytvořit soubor `/etc/xinetd.d/cvspserver`:

```
service cvspserver
{
 disable = no
 socket_type = stream
 protocol = tcp
 wait = no
 user = root
 server = /usr/bin/cvs
 server_args = -f --allow-root=/home/literakl/CVSROOT pserver
}
```

V obou případech parametr `--allow-root` ukazuje na CVSROOT, který bude takto zpřístupněn přes pserver. Pokud máte více repozitářů, můžete jej použít vícekrát.

V tuto chvíli je třeba ještě restartovat **inetd**, respektive **xinetd**. Pod RedHatem a spol. spusťte

```
$ /etc/init.d/inetd reload
```

respektive

```
$ /etc/init.d/xinetd reload
```

Nastavení CVSROOT

Nyní je třeba zinicializovat CVSROOT. Přihlásíme se jako uživatel vlastnící adresář CVSROOT, v tomto případě *literakl* a spustíme

```
$ cvs -d /home/literakl/CVSROOT init
```

Tím se nám vytvořil adresář `/home/literakl/CVSROOT/CVSROOT` obsahující spoustu administrativních souborů. Mezi nimi nám však chybí soubor `passwd`, který je určen právě pro vzdálený přístup. Jeho obsah je nápadně podobný systémovému souboru `/etc/passwd`. Každý řádek obsahuje jedno až tři políčka oddělené dvojtečkou. První políčko určuje přístupové jméno pro CVS. Tímto jménem se budete logovat do CVS. Druhé políčko určuje heslo zašifrované pomocí funkce `crypt`. A konečně poslední políčko určuje skutečného uživatele na systému. CVS bude provádět všechny změny jeho jménem.

Rozeberme si pravidla na praktickém příkladě:

```
$ cat /home/literakl/CVSROOT/CVSROOT/passwd
pub::pubcvs
literakl:HTRphPBvKJtjA:literaklliterakl:HTRphPBvKJtjA:literakl
oazanon:TXiF1923PHrtI:oazanon
```

První řádek určuje, že CVS uživatel *pub* nepotřebuje žádné heslo a je namapován jako systémový uživatel *pubcvs*. Takový uživatel většinou mívá přístup jen ke čtení. To snadno zajistíte, pokud vytvoříte skupinu *cvsusers*, do které přidáte uživatele *pubcvs*, *literakl* a *oazanon* a dáte této skupině právo čtení na celý `CVSROOT`.

```
$ groupadd cvsusers
$ vi /etc/group
cvsusers:x:503:pubcvs,literakl,oazanon
```

a v novém shellu

```
$ chmod -R g+r-w /home/literakl/CVSROOT
$ chgrp -R cvsusers /home/literakl/CVSROOT
```

Další řádky definují dva uživatele s heslem, kteří jsou namapováni na stejné uživatele, kteří již existují v systému. Takto jsme jim určili hesla pro CVS, která by měla být odlišná od systémových hesel.

Jenže jak ta hesla zašifrujeme? S Perlem je to hračka: (nejsem autorem)

```
$ vi cvspassword.pl#!/usr/bin/perl
srand (time());
my $randletter = "(int (rand (26)) + (int (rand (1) + .5) % 2 ? 65 : 97))";
```

```
my $salt = sprintf ("%c%c", eval $randletter, eval $randletter);
my $plaintext = shift;
my $crypttext = crypt ($plaintext, $salt);
print "${crypttext}\n";
```

```
$ chmod +x cvspassword.pl
$ ./cvspassword.pl heslo
```

Tímto končí konfigurace na straně serveru.

Klient

Uživatelé přistupující z cizího serveru musí nastavit proměnnou prostředí CVSROOT a zalogovat se. Je dobré uložit CVSROOT do souboru `~/.profile`, takže se nám vytvoří automaticky při přihlášení:

```
export CVSROOT=:pserver:literakl@localhost/home/literakl/CVSROOT
```

Zalogujeme se do CVS pomocí příkazu **cvs login**:

```
$ cvs login
Logging in to :pserver:literakl@localhost:2401/home/literakl/CVSROOT
CVS password: heslo
```

Pokud CVS neovládáte, mohu doporučit článek na Rootovi (*http://www.root.cz/clanek.phtml?id=1147*) a v Linuxových novinách (*http://www.linux.cz/noviny/2001-04/clanek08.html*).

Diskuse

Pavel Krebs: Přístup přes SSH je vcelku jednoduchý. Stačí nastavit dvě proměnné prostředí a je to. První proměnná je: `CVSROOT=:ext:server.domena:/adresar/na/serveru` druhá je: `CVS_RSH=ssh`.

Na serveru cvs nemusí běžet, jako při přístupu přes pserver. Dále je pak rozdíl, že se nepoužívá příkaz **cvs login** a **cvs logout**, ale heslo se zadává při každém příkazu. Chci-li si ušetřit zadávání hesla, musím pro ssh vygenerovat klíče a správně je nahrát do konfigurace ssh. Klíče se generují pomocí příkazu: `ssh-keygen -t rsa`. Tento příkaz vygeneruje dva soubory: `id_rsa` – privátní klíč, který je uložen na klientovi v adresáři `$HOME/.ssh` a `id_rsa.pub` – ten patří na server. Jeho obsah se přidá do souboru `$HOME/.ssh/authorized_keys`. Pak by vše mělo chodit bez otázky na heslo. Podobně se lze připojit i na CVS server z Windows, ale to už je trochu jiné téma.

Framebuffer v Linuxu

CIJOML

Úvod

Framebuffer – snad každý uživatel Linuxu toto slovo už slyšel, menší část však ví, o co se jedná a ještě menší část z nich tuto skvělou věc používá. Nastavení framebufferu opravdu není obtížné. Jeho konfiguraci vám předvedu na mojí prehistorické herní kartě – 3Dfx Voodoo 3 3000. Pro ty s větším množstvím peněz nebo silnějším hardware bude jediný rozdíl v tom, že v kernelu zvolí podporu pro framebuffer svojí karty. Takže se do toho pusťme, ať už si konečně neničíme oči díváním se na monitor s nízkou obnovovací frekvencí.

Nastavení jádra

Doporučuji využít kernel z poslední stabilní řady 2.4, i když framebuffer funguje i na řadách 2.2. Já použil jádro ze stabilní řady 2.4, konkrétně 2.4.19-pre10.

Vstoupíme do adresáře, kde máme jádro rozbalené a příkazem **make menuconfig** vyvoláme nabídku. Dále zvolíme:

```
Console drivers
Frame-buffer support --->
[*] Support for frame buffer devices (EXPERIMENTAL)
<*> 3Dfx Banshee/Voodoo3 display support (EXPERIMENTAL)
[*] Advanced low level driver options
<*> 8 bpp packed pixels support
<*> 16 bpp packed pixels support
<*> 24 bpp packed pixels support
<*> 32 bpp packed pixels support
[*] Select compiled-in fonts
[*] VGA 8x8 font
[*] VGA 8x16 font
```

Fonty a pixely jsou zakompilovány proto, aby po přechodu z XFree86 do konzole se "nerozsypalo" písmo v konzoli. Bez toho tato volba po spuštění XFree86 nefunguje. Pro ostatní karty je dobré je mít přeložené, ale pravděpodobně to nebude nutné pro správnou funkci vaší karty.

Po přeložení jádra, zapsání nového lila a rebootu bude konzole přepnuta automaticky do rozlišení 80x30, ale bohužel zase jen v rozlišení 60 Hz. To napravíme velice jednoduše po nahlédnutí do dokumentace jádra. Dle ní je nutné kernel instruovat už při startu jádra o přepnutí na vysoké rozlišení vložením příkazu přes parametr `append` v `/etc/lilo.conf`:

```
image=/boot/vmlinuz
label=linux
read-only
root=/dev/hda3
append="video=tdfx:1024x768-24@75"
```

Tím jsme docílili přepnutí konzole do rozlišení 1024x768 ve 24 bitové barevné hloubce a obnovovací frekvenci 75 Hz. Tato karta umí i mnohem vyšší obnovovací frekvence, narozdíl od mého monitoru. Konfiguraci pro váš monitor upravte dle jeho schopností.

Aby bylo možno tuto obnovovací frekvenci používat, je nutné upravit zdrojové soubory jádra. Jádro totiž standardně tuto obnovovací frekvenci neumí. Podporuje 76 Hz, se kterými můj monitor vždy po chvilce problikl. Úpravu jsem provedl v souboru `linux/drivers/video/modedb.c`. Můžete použít tento patch, který jsem již zaslal Rusty Russellovi. Doufám, že se objeví už ve verzi 2.4.19-rc2 a vy jej nebudete muset provádět.

```
--- linux/drivers/video/modedb.c Fri Dec 21 18:41:55 2001
+++ linux/drivers/video/modedb.c Sun Jun 30 12:55:59 2002
@@ -107,6 +107,10 @@
NULL, 70, 1024, 768, 13333, 144, 24, 29, 3, 136, 6,
0, FB_VMODE_NONINTERLACED
}, {
+ /* 1024x768 @ 75 Hz, 60.020 kHz hsync */
+ NULL, 75, 1024, 768, 12699, 176, 16, 28, 1, 96, 3,
+ 0, FB_VMODE_NONINTERLACED
+ }, {
/* 1280x1024 @ 87 Hz interlaced, 51 kHz hsync */
NULL, 87, 1280, 1024, 12500, 56, 16, 128, 1, 216, 12,
0, FB_VMODE_INTERLACED
```

Po startu jádra instruovaného o přepnutí na požadované rozlišení uvidíme tyto zprávy:

```
fb: Voodoo3 memory = 16384K
fb: MTRR's turned on
tdfxfb: reserving 1024 bytes for the hwcursor at d1818000
Console: switching to colour frame buffer device 128x48
fb0: 3Dfx Voodoo3 frame buffer device
```

Jak vidíte, obrazovka se přepnula do rozlišení 128x48. Na takovou obrazovku se nám velice pohodlně vejde celé základní menu konfigurace jádra bez nutnosti jeho skrolování, o možnosti sledovat filmy za pomoci mplayeru nebo televize skrze xawtv v konzoli ani nemluvě :-).

Další úpravy

Proberme si nyní, jakým způsobem je možné operovat s framebufferovou konzolí po nabootování systému. K tomu slouží program **fbset**. S jeho pomocí můžeme měnit rozlišení, obnovovací frekvenci i barevnou hloubku monitoru a umožňuje i další finesy. Pro bližší seznámení s tímto programem bych vás odeslal na manuálové stránky tohoto programu. My si zde probereme úplné základy.

Definice našich rozlišení jsou napsány v souboru `/etc/fb.modes`, ale jen pro 8bitové barvy. Proto jej musíme upravit tak, abychom dosáhli barevné hloubky, která nám bude vyhovovat. To provedeme tak, že pro rozlišení, co nás zajímá, upravíme vždy poslední číslo v druhém řádku na hloubku, jaké chceme dosáhnout.

Tento příklad by měl doufám stačit. Takto vypadá originální definice módu monitoru:

```
mode "1024x768-75"
  # D: 78.75 MHz, H: 60.023 kHz, V: 75.03 Hz
```

```
  geometry 1024 768 1024 768 8
  timings 12699 176 16 28 1 96 3
  hsync high
  vsync high
endmode
```

My jej upravíme takto:

```
mode "1024x768-75"
  # D: 78.75 MHz, H: 60.023 kHz, V: 75.03 Hz
  geometry 1024 768 1024 768 24
  timings 12699 176 16 28 1 96 3
  hsync high
  vsync high
endmode
```

Potom už jen nastavujeme konzoli tímto parametrem:

```
fbset -a 1024x768@75
```

To způsobí, že se naše konzole pro všechny textové konzole nastaví na 1024x768 při 75 Hz. Barevnou hloubku bude mít konzole takovou, jakou jsme doplnili do `/etc/fb.modes` – v mém případě 24 bitů. Informace o rozlišení, jaké vaše konzole právě používá, získáte příkazem **fbset -i**. Lahůdkou na dortu je možnost mít každou konzoli v jiné barevné hloubce a jiném rozlišení pouhým vynecháním parametru `-a` při volání **fbset**.

Diskuse

Petr Klíma: Pokud máte tu smůlu, že máte grafiku s čipem od nVidie a zároveň používáte ovladače do X přímo firemní (tedy z *http://www.nvidia.com*), je pro vás framebuffer v podstatě nepoužitelný. Můžete použít jedině VESA framebuffer, který ale neumí měnit frekvenci obrazu (používá natvrdo 60Hz) a je dosti pomalý. Už kvůli těm šedesáti hertzům je to dost pochybné řešení. Jestli vám nevadí, že konzoli nebudete provozovat přes framebuffer (přijdete o televizi v konzoli), můžete použít program svgatextmode. Více podrobností na *http://linuxdesktop.kn.vutbr.cz/linuxdesktop.php?link=117*, alias článek High-res konzole.

Pavel Trka: Článek dobrý, ale chtělo by to trošku víc informací, nakonec jsem je našel tady: *http://www.linuxhq.com/kernel/v2.4/doc/fb/* – doporučuji, je tam info o fb i na jiných kartách, nejen 3dfx ale i ATI, matrox… Pokud nemáte přístup k netu, tak to samé je přímo v dokumentaci k jádru (u mě `/usr/src/linux-2.4.18-3/Documentation/fb/`).

Linux – praktický průvodce

Stanislav Musil

Úvod

Předem musím říct, že kniha stojí za svým názvem praktický průvodce. Vše se vysvětluje opravdu na praktických příkladech. Není to jen suché omílání teorie, i když bez ní se to taky neobejde. Kniha začíná předmluvou samotného zakladatele Linuxu – Linuse Torvaldse. Ten zde píše o svém prvním setkání s počítačem a jak se dostal až k tvorbě Linuxu.

Obsah

Kniha je rozdělena na dvě větší části. První část je dělena na patnáct kapitol. Druhá část popisuje na praktických ukázkách 87 obslužných programů. Kniha obsahuje i čtyři dodatky (Regulární výrazy, Nápověda, Emulátory: Spouštění programového vybavení z jiných operačních systémů a Standard POSIX).

Začínáme s operačním systémem Linux

Kapitola je určena pro úplné začátečníky, kteří sedí poprvé před Linuxem. Seznámí je s přihlašováním a řekne, kdo to vlastně je ten "root". Naučí vás používat vestavěnou nápovědu. Ukáže jak vypsat adresář a jak pracovat se soubory. Autoři nezapomněli i na užitečné zkratkové klávesy, které značně urychlují práci v textovém režimu.

Úvod do obslužných programů

Tato kapitola je takový malý předúvod do druhé části. Jsou zde popsány ty nejzákladnější programy, se kterými se budete setkávat téměř na každém kroku. Jde o programy pro manipulaci se soubory, identifikaci a komunikaci s uživateli v systému.

Souborový systém

Dozvíte se, jak se vyznat v obrovské spleti adresářů, na kterou jste doposud nebyli zvyklí. Na konkrétní adresářové struktuře se dozvíte, co který adresář skrývá. Ale i to, jak používat přístupová práva k souborům a adresářům k tomu, abyste ochránili svá data před ostatními.

Příkazový procesor

Vysvětlí, jak je realizován standardní vstup, výstup, přesměrování, roury a obecné použití speciálních znaků v příkazových procesorech.

Grafické uživatelské rozhraní

Poměrně krátká kapitola, která se věnuje grafickému rozhraní Linuxu. Je zde stručně pohovořeno o systému X-window a o správcích oken tohoto systému. Podrobněji je popsán správce oken fvwm.

Práce v počítačové síti

Začíná základními pojmy a postupně se dostáváte z práce v místní síti až na síť Internet. Ukáže vám základní práci s telnetem, rsh, ftp. Dozvíte se i něco o DNS a NIS.

Editor vi a emacs

Tyto kapitoly mají výukový charakter, kdy se seznámíte s obsluhou celoobrazovkového editoru vi a Emacs. V závěru každé z těchto kapitol je seznam příkazů pro práci s editorem. Navíc u Emacsu je popsána i spolupráce se systémem X-Window.

Příkazové procesory bash, TC shell, zshell

Dopodrobna rozepsaná práce s jednotlivými příkazovými procesory. Jejich přednosti a zápory. Vyzdviženy jsou možnosti použití příkazového procesoru jako programovacího jazyku. Vše je řádně podpořeno spoustou praktických ukázek, které lze využít i v praxi.

Programovací nástroje

Jedná se o nástroje pro práci se soubory jazyka C. Tato jediná část knihy je spíše pro pokročilé uživatele a programátory.

Správa systému

Poslední kapitola první části je určena spíše pro ty, kteří mají práva super uživatele. Dozvíte se něco o zálohování, správě uživatelů, kontrole log souborů, rozdělování disku a zavádění systému.

Každá kapitola je zakončena souhrnem toho, co jste probrali a opakovacími cvičeními. Každé cvičení je rozděleno do dvou částí (standardní a pro pokročilé). Pokud zvládnete po přečtení každé kapitoly odpovědět na opakovací cvičení, tak si můžete říci, že jste danou kapitolu zvládli.

Na zhruba 360 zbývajících stranách budete podrobně seznámeni s 87 základními programy pro práci se soubory, prací v síti, programovacími nástroji, programy pro správu zdrojových kódů a dalšími pomocnými, ale i mocnými obslužnými programy. Každý program je uveden stručným popisem, za kterým následuje syntaxe programu. Dále pak argumenty, volby, poznámky a praktické příklady, na kterých je daný program předveden a vysvětlen. Vyjmenovávat je nemá smysl, ale uvedu jen pár: **dd**, **tar**, **rsh**, **pine**, **nohup**, **fsck** a další.

Dodatek A se věnuje regulárním výrazům. Dodatek B obsahuje nápovědu formou 20 otázek a odpovědí na nejčastější dotazy začínajících uživatelů. Pro ty, kteří ještě trošku závisí na programech běžících pod operačním systémem DOS a Windows společnosti Microsoft, je věnován dodatek C. Ten obsahuje stručný popis a použití DOS emulátoru dosemu a programu Wine, který umožňuje spouštění některých aplikací pro Windows. Dále je tu zmínka i o executoru, ten umí spouštět aplikace pro OS Macintosh. A nechybí i iBCS, což je rozšíření jádra Linuxu umožňující spouštět programy přeložené pod SCO UNIX na počítačích Intel x86. Ovšem celý dodatek C se tomuto věnuje opravdu jen velice okrajově. Nechybí ani slovník pojmů a rejstřík, bez kterého by hledání v tak velké knize asi nebylo možné. Poslední dodatek D se věnuje standardu POSIX.

Závěrečné hodnocení

Kniha je určena především začínajícím a mírně pokročilým uživatelům, kteří se nebojí trochu zaexperimentovat. Je přímo prošpikována praktickými příklady, které pomohou k pochopení práce jednotlivých programů. A pomohou vám při konfiguraci a správě systému. Knihu vám mohu jen doporučit. Cena knihy je myslím přiměřená vzhledem k množství stran a kvalitě obsahu. Co bych vytkl je spíše droboulinká chybička, kterou lze přehlédnout. A to že u některých slov použitých např. u popisek obrázků je použit jiný než český font. A slova jsou špatně čitelná. Jedná se opravdu jen o jednotlivá slova.

Název knihy: Linux – praktický průvodce, Autor: Mark G. Sobell, Vydalo: Computer press
Doporučená cena: 745Kč / 1073SK, 946 stran

Jaderné noviny

Leoš Literák

Status VM a IDE v řadě 2.5

Linus Torvalds oznámil:

"Nedávno se v řadě 2.5 událo spoustu věcí, o čemž svědčí slavné detaily v Changelogu, ale chtěl bych vyzdvihnout 2.5.14, protože se v ní zásadně změnil způsob správy dirty stavu ve správě virtuální paměti. Zásluhu na tom má Andrew Morton. Kód byl nejen výrazně vyčištěn, ale za určitých okolností běhá podstatně rychleji. V řadě 2.5 je spousta jiných změn, ale žádná není tak zásadní. Prosím otestujte ji (ale buďte opatrní a udržujte zálohy)."

Bert Hubert se zeptal, čí verze virtuální paměti je použita: ta od Rika van Riela nebo od Andrea Arcangeliho? Andrew vysvětlil, že VM je široký pojem. Aktuálně je použita verze od Andrea začleněná do hlavního jádra ve 2.4. Andrew jen naučil alokátor stránek, že všechny dirty paměti se zapisují skrze stránky a ne někdy přes stránky, jindy přes buffer. Přidal navíc clustering writeback.

V řadě 2.4 jsou dirty data skrze systémovou funkci write(2) zaobalena do `buffer_heads` a umístěna do globální bufferu, kde čekají na zapsání. A dirty data ze sdíleného mapování [shared mappings] jsou připojená ke svému inodu. Kdežto v 2.5 byl buffer odstraněn a dirty data z write(2) jsou zpracovávána stejně jako dirty data z mmap(). Dále byly upraveny **kupdate** a **bdflush**. Vše je nyní stránkově orientováno.

Martin Dalecki zaslal tunu patchů do IDE kódu a vývojáři začali debatovat nad technickými body. Najednou Anton Altaparmakov vypěnil: "Od té doby, co jsi novým správcem IDE, jsme tě viděli pouze odstraňovat jednu funkci za druhou ve jménu vyčištění kódu bez toho, aby jsi za ně uvedl nějakou adekvátní náhradu. Všechny kritické opravy zaslali jiní lidé, což nevzbuzuje v tebe důvěru. Dokonce i Alan Cox před časem napsal, že ti nedůvěřuje."

Ale Linus se Martina zastal:

"Koho to zajímá? Našel jsi cokoliv, co Martin odstranil a mělo nějakou cenu? Určitě ne. Pánové, měli byste si uvědomit, že IDE vrstva má v sobě OSM LET naprostých zmetků. Nikdy to nebylo pročištěno. Má to v sobě tak odporné věci, až je to příšerné. Podle mě je mnohem jednodušší přidávat novou funkčnost do čistého kódu."

"Veškeré informace z `/proc/ide` jsou dostupné přes **hdparm** a pro vaše drahé embedded systémy zřejmě zabírají méně prostoru, když jsou v uživatelském prostoru. Tak kde je problém?"

Alan Cox ale poznamenal, že `/proc/ide` obsahuje užitečné informace, které jinde nejsou dostupné. Například jaký řadič řídí disky, které disky jsou dostupné, apod. Ale Linus napsal, že by raději viděl někoho přidat tato zařízení do skutečného stromu zařízení, kde by tento typ informací byl velmi viditelný. Zatím devicefs není automaticky připojen, ale jedná se o jediný skutečně univerzální způsob prezentace takovýchto informací.

Podpora velkých souborů

Peter Chubb oznámil: "V současnosti je linux omezen 2 TB souborovým systémem a to dokonce i na 64-bit systems, protože je zde spousta různých míst, kde je blokový ofset nastaven na unsigned nebo int (32-bit proměnné)". Zaslal link na patch (*http://www.gelato.unsw.edu.au/patches/2.5.14-largefile-patch*), který implementuje nový typ `sector_t`, který má držet hodnotu ofsetu u sektorů a bloků. Dodal, že na svém starém pentiu dokázal vytvořit 15 TB velký souborový systém připojený jako JFS skrze loop zařízení – a zdá se, že vše funguje. Existuje pár programů z uživatelského prostoru, které je třeba opravit (**parted**, **mkfs**).

Andrew Morton odpověděl: "Hlasuji pro začlenění. 2 TB zabraňuje některým lidem používat řadu 2.4. Řada 2.6 bude zřejmě potřebovat 64-bit čísla. Další překážkou budou indexy stránkové cache. Tam bude limit buď 8 TB nebo 16 TB, v závislosti na znaménku."

Různým lidem se patch také líbil a Martin Dalecki napsal, že jeho části začlení. Později Peter oznámil novou verzi pro kernel 2.5.15 a Christoph Hellwig napsal, že patch vypadá velmi dobře a že doufá, že bude brzy začleněn.

Zpětná kompatibilita

Během diskuse nad novými patchi pro diskové quoty Linus Torvalds navrhnul odstranit určitý kód, který byl vytvořen pro poskytování zpětné kompatibility. Zeptal se, zda existuje nějaký důvod používat tento kód, když stačí přejít na novou verzi nástrojů pro quotu. Alan Cox reagoval s tím, že většina lidí už stejně používá řadu 2.4 a quota nástroje s podporou 32bitových UID, takže se nic moc nestane. Jan Kara tedy napsal, že zašle patch na odstranění tohoto kódu.

Martin Dalecki navrhnul rovnou odstranit i IPC_OLD. Jenže Alan odpověděl, že tento kód nezabírá skoro žádný prostor, ale zajišťuje, aby i staré XFree86 běželo na nových jádrech. Tak proč jej odstranit. Podle Martina je iluzí myslet si, že je možné spouštět tak staré a.out binární soubory na moderním jádru. Christoph Hellwig jej však vyvedl z omylu: "Samozřejmě že můžeš. Dokonce i nejnovější verze OpenLinuxu s jádrem 2.4.13-ac používá instalaci založenou na libc4/a.out kvůli nárokům na prostor. A nezapomeňte na staré binárky quake1, kterého si čas od času zahraji :)". Martin byl ohromen tím, že to funguje a Alan mu doporučil, ať si příště nejdříve věci ověří, než něco navrhne odstranit. Pak dodal, že máme 100% kompatibilitu až do libc 2.2.2. Starší binárky pravděpodobně nebudou fungovat, neboť jsme odstranili některá obskurní systémová volání.

V tu chvíli Christoph poznamenal, že plánuje pro řadu 2.5 nechat zastaralá systémová volání záviset na CONFIG_COMPAT_*, což umožní jak vytvořit obrovský zpětně kompatibilní kernel, tak i malý kernel bez ní. Ve skutečnosti je v jádře spousta kódu, který by mohl být pro moderní instalace odstraněn. Martin i Alan schválili tento nápad.

Diskuse nad patenty v RTLinuxu

Během diskuse někdo zmínil, že algoritmy použité v RTLinuxu jsou patentovány. Karim Yaghmour napsal:

"Už dva roky válčím proti tomuto patentu. Během této doby jsem se setkal s mnoha lidmi a bavil se s nimi o něm. Dnes jsem si jist, že rtlinux patent brání Linuxu proniknout do této oblasti. Linux nikdy nebude životaschopný embedded operační systém, dokud nás někdo nezbaví tohoto patentu. Mnoho firem se rozhodlo použít raději WinCE kvůli němu. Mnoho vývojářů naší komunity kolem real time Linuxu je znepokojeno faktem, že kerneloví vývojáři považují real-time za vedlejší trh."

"Poslední průzkumy VDC *http://www.linuxdevices.com/articles/AT6328992055.html* ukazují, že hlavním důvodem, proč se Linux neprosazuje do embedded oblasti, je ve skutečnosti real-time. Proto se nedivme, že zavedení dodavatelé (WindRiver, QNX, ..) se necítí Linuxem ohroženi. Ví, že kdykoliv bude Linux vyhodnocován, bude nakonec zavrhnut kvůli tomuto patentu. A protože embedded zařízení mají předhonit desktopy a servery v počtech nasazení, myslím, že se jedná o více než okrajový trh. Dokud nebude patent na RTLinux zrušen, Linux nepronikne výrazněji do embedded zařízení. Bohužel."

Linus Torvalds odpověděl: "Tento patent je ale licencován pro GPL kernely, tedy i Linux. Viz *http://www.fsmlabs.com/about/patent/openpatentlicense.htm*. Slyšel jsem hodně diskusí na téma, že patenty s licencí pro GPL software podvracejí patentový systém stejně, jako copyleft podvrací copyright. Dokonce i FSF podporuje tento patent. V podstatě, pokud půjdeš příliš daleko a odejdeš od GPL, ztratíš také patentová práva. Ale pokud zůstaneš na správné straně, licence je tvoje a zadarmo. Přesně jak GPL požaduje."

Karim odpověděl: "Rozumím tomu, co říkáš, ale velkou část historie patentu neprobíhala skutečná komunikace s kernelovými vývojáři. Pokusím se vysvětlit své stanovisko. Když Jerry Epplin získal patent na začátku roku 2000, poslal do konference dotaz, ve kterém jasně říká, že jeho hlavním účelem je obrana. A tak real-time komunita čekala, co bude následovat. Pak přišla první verze licence, která porušovala GPL tím, že požadovala registraci všech uživatelů ve FSMLabs. Jedna z podmínek říkala, že software je zdarma pro programy šířené pod GPL kdekoliv a všechny uživatele RTLinuxu (nezávisle na licenci). Victor tím říkal, že pokud nechcete platit licenci, buď uvolněte software jako GPL nebo používejte jeho Open RTLinux Execution Environment."

"Toto trvalo, dokud FSF veřejně neprohlásila, že patent porušuje GPL. Eben Moglen posléze zaslal opravenou licenci a její důsledky: *http://www.aero.polimi.it/ rtai/documentation/articles/moglen.html*. Pak se situace zklidnila a vývojářský tým RTAI se snažil plnit Ebenovy doporučení. Vše by bylo v pořádku, kdyby Victor nevnesl další nejistotu: *http://linuxdevices.com/articles/AT6164867514.html*. Eben říkal, že real-time aplikace nejsou předmětem patentu, jenže najednou Victor Yodaiken napsal, že se patent vztahuje na veškerý software. Tímto Victor zpochybnil závěry respektovaného právníka Ebena Moglena."

"V současnosti RTAI jednoznačně vede nad RTLinuxem. Abyste rozuměli, co se děje. Spousta vývojářů měla zájem přispívat i do RTLinuxu, ale Victor je všechny odradil. Je pro to logický důvod: FSMLabs používá duální licenci RTLinuxu a poskytuje svým klientům uzavřenou licenci. To znamená, že všechen kód musí vlastnit FSMLabs. Pokud se podíváte do jejich zdrojových kódů, všude je copyright pouze FSMLabs. Na tom není nic špatného, ale pak jsou všechny nezávislé příspěvky zahazovány. V tuto chvíli nejnovější vývoj RTLinuxu není dostupný jako GPL a musí být zakoupen. To není problém, neboť RTAI převyšuje RTLinux ve schopnostech, portech i podpoře. Problémem je, že RTLinux patent je používán jako FUD proti RTAI. A když se někdo podívá na RTAI, narazí na FUD a nakonec skončí u jiného OS."

Linus však nesouhlasil: "Musí být část poskytující RealTime GPL? Ano. Stejně jako kernelové moduly, pokud jsou odvozenou prací, což je v tomto případě zřejmé. Tak rozděl svůj problém na RT ovladač a aplikace a přestaň s tímto hloupým FUD. Znechucuje mě používání patentu jako falešné stopy. Skutečným problémem je, že některým lidem se nelíbí, že je kernel uvolněn jako GPL. Přestaň dělat vyjímky. Osobně jsem šťasten, že lidé mají další důvod uvolňovat své moduly jako GPL. Vidím příliš mnoho problémů s věcmi jako je ovladač od firmy nVidia a když vidím, že někoho děsí GPL, tak mě to nezajímá. Někteří lidé (ty a Karim) si myslíte, že požadavek GPL ubližuje Linuxu v oblasti embedded. Totéž ale tvrdili lidé z BSD o Linuxu na serverech a desktopech. Osobně sázím na Open Source. I v embedded. A pokud s tím někdo nesouhlasí, tak mě to nezajímá."

Adeos, nanokernel běžící s Linuxem

Karim Yaghmour oznámil první verzi nanokernelu Adeos. Tisková zpráva je dostupná na adrese *http://www.freesoftware.fsf.org/adeos/pr-2002-06-03.en.txt* a projekt na *http://freesoftware.fsf.org/projects/adeos/* . Kód je uvolněn pod licencí GPL. Adeos je založen na výzkumu a publikacích z raných devadesátých let. Místo, aby nejdříve postavili nanokernel a pak pro něj postavili klientské operační systémy, tak použili funkční OS a vložili před něj nanokernel. Nyní je možné vložit další operační systémy vedle Linuxu.

Adeos umožňuje existovat více doménám vedle sebe, přičemž žádná doména nevidí jinou doménu, ale všechny vidí Adeos. Doména je obvykle kompletní operační systém, ale neděláme v tomto ohledu žádné předpoklady, co přesně doména je. Aby bylo možné sdílet hardware mezi různými operačními systémy, Adeos implementuje rouru přerušení [interrupt pipeline] – ipipe. Každá doména má vstup do ipipe. Každé přerušení je vloženo do ipipe a tak je propagováno ke všem doménám. Místo povolení/zakázání přerušení domény manipulují s rourou. Pokud je prvek v rouře uzamčen [ipipe stage is stalled], pak doména neobdrží žádné přerušení. Každý prvek může samozřejmě dělat s přerušením spoustu věcí, mezi nimiž je rozhodnutí nepropagovat přerušení dále. Nezávisle na operacích provedených v rouře, Adeos sám o sobě si nehraje s maskou přerušení. Jedinou vyjímkou je přidávání/odebírání domény z ipipe. To také znamená, že žádný OS nesmí používat skutečné hradwarové CLI/STI (stejnou funkčnost nabízí funkce odemykání/zamykání [stall/unstall].

Jejich práce je postavena na těchto výzkumech:

> D. Probert, J. Bruno, and M. Karzaorman. "Space: a new approach to operating system abstraction." In: International Workshop on Object Orientation in Operating Systems, pages 133-137, October 1991.
>
> D. Probert, J. Bruno. "Building fundamentally extensible application-specific operating systems in Space", March 1995.

D. Cheriton, K. Duda. "A caching model of operating system kernel functionality". In: Proc. Symp. on Operating Systems Design and Implementation, pages 179-194, Monterey CA (USA), 1994.

D. Engler, M. Kaashoek, and J. O'Toole Jr. "Exokernel: an operating system architecture for application-specific resource management", December 1995.

Některé možné příklady použití:

Podobně jako User-Mode Linux, je možné spustit dva linuxové kernely na jednom hardwaru. Narozdíl od UML, kde jeden kernel běží na druhém, pod Adeosem budou běžet skutečně vedle sebe. Tak by bylo možné říci, aby jeden kernel používal spodní půlku RAM a druhý horní během bootu. Zjistili jsme ale, že by bylo třeba provést hodně změn, například aby jen jeden inicializoval hardware. Nicméně tato možnost by měla být lépe prostudována.

Podobně by mohly běžet vedle Linuxu další kernely a BSD je první kandidát. Bylo by hezké sledovat virtuální stroje jako je VMWare a Plex86.

Začleněním Adeose do hlavního stromu jádra by se vyřešil hlavní problém s kernelovými debuggery: skákání do přerušení v jádře. Debugger by pak mohl mít podobu modulu a nevyžadoval by žádné speciální patche.

Ovladače, které požadují absolutní prioritu a nemají rády, jak ostatní části kernelu zacházejí s CLI/STI, si mohou vytvořit vlastní doménu a umístit se v ipipe před Linux. Tento přístup poskytuje mechanismus pro poskytování garantovaných realtime odpovědí [responses].

Šetření spotřeby baterií

Andrew Morton zaslal patch, jehož cílem je ušetřit energii baterek na laptopech. Patch najdete na adrese *http://www.zip.com.au/~akpm/linux/patches/2.5/2.5.20/pdflush-sysctl.patch*. Myšlenka je jednoduchá: nastavením hodnoty proměnné `laptop_mode` na 1 v `/proc/sys/vm/` se jádro přepne do módu určeného pro notebooky a laptopy. Tento mód se snaží minimalizovat frekvenci buzení disku. Data na zapsání [dirty data] zůstávají v paměti delší dobu než normálně. Pokud je disk z nějakého důvodu (například čtení) vzbuzen, pak se toho využije a veškerá dirty data jsou zapsána. Pokud se kernel rozhodne zapsat nějaká data na disk, tak zapíše všechna data. Perioda zápisu dat je nastavitelná přes proměnnou `laptop_writeback_centisecs` a předvolená hodnota činí pět minut.

Rozběhla se diskuse jak nad dalšími vlastnostmi, tak nad technickými detaily. Andreas Dilger podotkl, že pětiminutový interval není vhodný, neboť koliduje s nastavením šetřícího módu v BIOSech. Proto doporučil nejméně 15-20 minut. Andrew tuto námitku přijal a nastavil proměnnou na dvacet minut.

Zkracování ext2 a ext3 adresářů

Někdo připomenul odvěký problém, že po smazání souborů a adresářů z některého adresáře se bloky alokované pro adresář neuvolní. Odesílatel znal, že řešením je vytvořit nový adresář, zkopírovat do něj obsah starého adresáře, smazat jej a přejmenovat nový adresář. Toto se mu však zdálo ošklivé a tak se zeptal, zda existuje jiný způsob, jak zkrátit [shrink] adresář. Andreas Dilger odpověděl, že takováto vlastnost by vyžadovala spoustu práce, nicméně Stephen C. Tweedie napsal, že Daniel Phillips plánuje začlenění HTree pro rychlou indexaci adresářů pro ext2/3.

Alexander Viro měl za to, že limitovaná verze zkracování by neměla být složitá a hned načrtl způsob, jak na to a nabídl se, že ji napíše, což původního odesílatele velmi potěšilo. Stephen však našel pár slabších míst tohoto návrhu. Andrew Morton se vložil do diskuse s tím, že právě vložil podporu pro HTree pro ext3 do jádra 2.5.23. Diskuse pak pokračovala nad technickými detaily implementace.

Stav capabilities

Michael Kerrisk se zeptal, jak pokračují capabilities souborového systému v řadě 2.4. Věděl o částečné podpoře od řady 2.2 a zajímalo ho, zda se nějak pokročilo ve vývoji. Dax Kelson odpověděl, že VFS v řadě 2.5 podporuje rozšířené atributy (od 2.5.3). Plán byl ukládat do nich capabilities. Takže když už existuje infrastruktura, někdo musí:

- definovat formát uložení capabilities
- naučit `fs/exec.c` používat capabilities uložené se souborem
- napsat **lscap(1)**
- napsat **chcap(1)**
- auditovat/opravit všechny SUID root programy, aby používaly capabilities
- Nastavit správné capabilities pro každý z nich s **chcap(1)**

```
# find / -type f -perm -4000 -user root -exec chmod u-s {} \;
```

- Začít slavit

Ale Jesse Pollard zaslal odkaz na *http://lsm.immunix.org/* a řekl, že kromě posledního bodu je již vše hotovo v rámci projektu Linux Security Modul. Chris Wright upřesnil, že projekt LSM podporuje capabilities přesně tak, jak jsou v jádře a používání rozšířených atributů tedy chybí. Patche jsou vítány.

Jádro 2.5.25

Linus Torvalds oznámil jádro 2.5.25. Mezi hlavní změny patří začlenění patchů pro PPC, SCSI, USB, kbuild a ovladačů vstupních zařízení [input drivers]. Dále toto jádro obsahuje podporu pro vnitřní časovač [internal kernel times] neběžící na 100 Hz na platformě x86. Nyní je frekvence nastavena na 1KHz, ale pokud by někdo potřeboval, mohla by se tato hodnota nastavit při konfiguraci jádra. Otevřeným úkolem je sjednocení pojmenování diskových zařízení, když teď IDE i SCSI začínají používat DriverFS. Cílem je mít čistý způsob přístupu k disku, při kterém se nemusíme starat, zda je disk připojen přes IDE či SCSI nebo cokoliv jiného.

Matthias Andree se zeptal, zda se chystají hoši od LVM opravit jeho nefunkčnost [breakage] v řadě 2.5. Nebo zda jeho náhradou v řadě 2.5 bude EVMS. Joe Thornber odpověděl, že Heinz Mauelshagen udržuje LVM 1.0 v řadě 2.4. Jedná se jen o opravy, nové funkce nejsou přidávány. Lasdair Kergon, Patrick Caulfield a Joe pracují na mnohem obecnějším ovladači mapujícím zařízení pro řady 2.4 a 2.5. Ovladač je podle něj již velmi stabilní. Vývojáři nemají zájem o udržovaní LVM1 v řadě 2.5 kvůli jeho špatnému designu – a nechtějí plýtvat svými zdroji. Alexander Viro tedy navrhnul, ať je LVM1 kompletně odstraněno z řady 2.5, když nefunguje a nikdo se o něj nestará. Žádná odpověď však nepřišla.