

Linux prakticky ako server/ 11.časť

V minulých častiach sme sa venovali skôr teoretickým oblastiam a vnútornej podstate Linuxu. Uznávam, že boli trochu nezábavné a viac teoretické ako praktické, ale uvidíme ďalej, že naše vedomosti v budúcnosti zúročíme. Dnes sa už naozaj, ale naozaj budeme venovať praktickej časti. Dlhý som rozmýšľal, čo by tak na úplnom začiatku výstavby servera malo byť urobené ako prvé. Po určitých skúsenostiach som dospel k tomu, že ako prvé si na našom linuxovom serveri sprevádzkujeme zapisovaciu / prepisovaciu CD mechaniku – jednoducho nazývanú napáľovačku.

Možno teraz niekto namietne, že sa opakujem, veď napáľovačka – zapisovačka – prepiskovačka sa už preberala nedávno v časti **Linux ako desktop**. No to je síce pravda, ale na desktope sa narába s týmito zariadeniami tak akosi „konzumne“ – jednoducho sa iba používa. My si však neskôr ukážeme inú – kreatívnejšiu činnosť s napáľovačkou.

Druhý argument je, že na pravom serveri sa nepoužíva systém X Window, a preto nemôžeme použiť žiadny oknoidný grafický napáľovací program. Preto sme odsúdení k príkazovému riadku. A verte mi, tak sa nám to zapáči, že to budeme používať aj vtedy, keď budeme môcť použiť aj grafické rozhranie.

Trochu teórie

Ešte predtým, než sa pustíme do samotného napáľovania, vysvetlíme si dôležitú teóriu. A zatiaľ čo desktopistom stačí len to najnutnejšie, my sa teórii napáľovania pozrieme na zuby lepšie.

Rozdelenie CD médií

Tak ako to už v slovenčine býva, veľmi často sú zaužívané niektoré výrazy, ktorých význam nie vždy je správny, alebo má ešte niekoľko iných významov. Ako príklad môžeme použiť slovo „cédečko“. Všetci vieme, že pochádza zo skratky CD – Compact Disk. Jeho slovenský preklad - kompaktný disk – sa v praxi naozaj neujal. Jeho prvotný význam označuje médium – teda ten lesklý guľatý kruh, čo ho niektorí „experti“ nosia na šnúrke okolo krku alebo na spätnom zrkadle v aute. Bohužiaľ slovom „cédečko“ často mienime aj CD mechaniku, teda zariadenie na čítanie spomínaného média. My však budeme dôsledne dodržiavať názvoslovie, ktoré si tu vysvetlíme.

CD disky sa delia na:

- Ø CD-ROM
- Ø CD-R
- Ø CD-RW
- Ø CD-RWM

CD-ROM (*Compact Disk – Read Only Memory* = disk - pamäť iba na čítanie) je disk určený iba na čítanie, pričom počet čítaní nie je fyzikálne obmedzený. Takto sa označujú disky, ktorých obsah je vytvorený vylisovaním z matrice vo výrobe. Jedná sa o tie médiá, ktoré sa najčastejšie nachádzajú v rôznych časopisoch alebo ako nosič ovládačov rôznych počítačových komponentov a ako zdroj distribuovaného programu. Obsah tohto typu média nemožno meniť. Výroba CD-ROM disku je v domácich podmienkach v podstate nemožná, ale existuje dostatok firiem, ktoré nám na objednávku z určitého vzoru vyrobia žiadané množstvo nahraných diskov, pričom cena sa odvíja od počtu objednaného množstva, počtu farieb použitých pri potlačí a spôsobu obalovania.

CD-R (*Compact Disk Recordable* = zapisovateľný disk) je disk, na ktorý je povolený iba jeden zápis. Médium sa dodáva v „surovom“ stave a na zapisovacej mechanike sa do neho nahrávajú príslušné dáta. Po nahraní dát sa disk „uzamkne“ (teda natvrdo sa definuje koniec dát) a tento disk je použiteľný iba na čítanie, pričom počet čítaní nie je fyzikálne obmedzený. V prípade, že sme sa pri tvorbe dát pomýlili alebo počas zapisovania došlo k nejakej chybe alebo poruche, médium je (spravidla) nepoužiteľné. Niektoré druhy CD mechaník umožňujú dopisovanie CD-R média. To značí, že ak sme pri prvom zapisovaní dát nezaplňli médium úplne, môžeme na nevyužitú miesto neskôr ešte doplniť ďalšie dáta. Na budúce dopisovanie dát však musíme myslieť už pri prvom zapisovaní a vtedy nesmieme médium „uzamknúť“. Musíme mať na pamäti, že pri každom dopisovaní, ktorému hovoríme *session* sa okrem dát musí na disku vytvoriť nová tabuľka súborov, ktorej sa hovorí **TOC tabuľka** (*Table Of Contents*) a tá ukrojí okolo 22,5 MB priestoru! Takže ak chceme dopísať súbor o veľkosti pár kilobajtov, zápis nám „zožerie“ asi 23 MB!

Možnosť dopisovať na CD-R médiá sa označuje **multi-session**.

A aké je praktické využitie?

Môj linuxový server si sám pravidelne zálohuje svoje konfiguračné súbory a dáta, uložené v databáze. Keďže ich veľkosť je do 20MB, bolo mi ľúto kvôli tomu denne mŕňať jedno médium – zvlášť v dobe, keď médium stálo vyše 200 (!) Sk! Preto som využíval *multi-session*, aby som médium využil naplno.

CD-RW (*Compact Disc Rewritable* = prepisovateľný disk) je disk, na ktorý je umožnený viacnásobný zápis a výmaz. Na takéto médium je možné niečo zapísať, potom to zmazať a znova zapísať a to niekoľkonásobne za sebou. Zatiaľ čo počet zápisov a výmazov je fyzikálne limitovaný okolo počtu 1000 krát, čítanie je tak ako u predchádzajúcich diskov neobmedzené. Na zápis a výmaz CD-RW médií musí byť uspôsobená aj mechanika – tzv. prepisovačka (zapisovačka nestačí, aj keby sme na CD-RW médium chceli zapísať iba raz!). Keď sa vrátim k môjmu serveru, isto vás napadlo, že už *multi-session* na CD-R disky nepoužívam. Využívam CD-RW mechaniku a médiá, kde niečo zapíšem, na druhý deň médium zmažem a zapíšem nové dáta a tak dokola.

Dohodnutá norma zaručuje (teda – mala by!), že médiá, zapísané v iných mechanikách by mali byť čitateľné v ostatných mechanikách. Napríklad, ak vytvorím CD-RW cédečko v prepisovacej mechanike, mal by som ho prečítať aj v obyčajnej CR ROM mechanike tak, ako bežné lisované médium.

Dá sa povedať, že to v praxi aj funguje, ale stane sa, že niektorá staršia mechanika nechce prečítať CD-R alebo CD-RW médium.

Pre záznam na CD-R a CD-RW médiá sa spravidla používajú tie isté programy a utility k tomu určené.

CD-RWM (*Compact disk – Read Write Memory*), niekedy označované aj *CD-RAM* je taký disk, ktorý sa správa ako bežná disketa alebo pevný disk v počítači. Do prepisovačky zasunieme médium, príslušne ho naformátujeme a potom môžeme naň zapisovať tak ako keď kopírujeme na disk, prípadne môžeme z neho súbory zmazať. Bohužiaľ – to zvláštne formátovanie je veľmi neštandardné a tak údaje takto zapísané sa nedajú prečítať v inej mechanike, len v tej v ktorej boli vytvorené. Preto sa týmto typom médií nebudeme v Linuxe zaoberať.

Je dobré vedieť, že štandardy tvorby jednotlivých spôsobov zápisu dát sú vedené vo „farebných“ knihách. Existuje *Red Book*, *Yellow Book*, *Orange Book*, *Blue book* alebo *White Book* a iné, ktoré predstavujú akési normy, záväzné pre všetkých výrobcov médií a mechaník.

Rozdelenie CD mechaník

Podobne ako médiá, tak aj CD mechaniky sa delia na čítačky, zapisovačky alebo prepisovačky. Už zo slangového výrazu je zrejmý význam jednotlivých typov.

Okrem spôsobu práce sa mechaniky delia podľa spôsobu pripojenia k ostatným komponentom v počítači – spravidla k základnej doske.

Podľa spôsobu pripojenia poznáme mechaniky:

- Ø IDE
- Ø SCSI
- Ø USB
- Ø 1394 (FireWire)
- Ø PCMCIA
- Ø na paralelný port

V podstate všetky typy je možné pod Linuxom sprevádzkovať, niektoré s určitými obmedzeniami. Okrem IDE a SCSI zariadení sú tie ostatné trochu exotické a pomerne veľmi drahé. Preto si ich dnes nebudeme všímať, čo povieť?

V minulosti existovali iba SCSI mechaniky a preto bola v Linuxe vyvinutá aj patričná podpora. Bohužiaľ, SCSI mechaniky boli a sú stále veľmi drahé a preto, keď sa na trhu objavili dnes už klasické a podstatne lacnejšie IDE/ATAPI CD mechaniky, nastala snaha nahradiť SCSI systémom IDE a tak vznikla potreba vyriešiť aj ich podporu v Linuxe. Možností je niekoľko, ale najspoľahlivejšie riešenie je metóda zasielania SCSI príkazov cez IDE kanál k mechanike (trochu nesprávne sa to v linuxovej brandži nazýva *IDE-SCSI emulácia*, ale nenašiel som jednoduchý výraz, ktorý by vystihoval podstatu, preto aj my sa budeme dopúšťať tejto nepresnosti a budeme používať tento pojem). V skutočnosti sa v Linuxe využilo už to dobré a funkčné, všetky spoľahlivé programy a utility na vypaľovanie, len sa dorobil modul do jadra, ktorý zabezpečil komunikáciu s IDE mechanikami.

A práve tomuto – dnes najčastejšiemu riešeniu sa budeme venovať.

Takže otázka znie: *Akú mechaniku v Linuxe použiť?*

A odpoveď je: Bežne dostupnú zapisovaciu (alebo najlepšie prepisovaciu) CD mechaniku, ktorá sa pripája na radič IDE. To je taký istý radič, na ktorý sa pripája aj pevný disk v počítači. Vieme, že v bežnom počítači bývajú dva radiče IDE, kde na každý z nich je možné pripojiť dve IDE zariadenia, teda celkovo 4 zariadenia, najčastejšie pevné disky a CD mechaniky. Takže nájdeme jednu voľnú „kšandu“ a využijeme ju.

Pozor!

Pokiaľ nám to hardvérová konfigurácia v našom počítači dovoľuje, doporučuje sa pripojiť CD mechaniku na sekundárny radič, teda na iný kábel, ako je zapojený harddisk. Je to z dôvodu zabezpečenia toku dát.

Nastavenie jadra a systému

Prvou našou úlohou je presvedčiťadlo jadro Linuxu, aby CD prepisovacia mechanika, ktorá je v skutočnosti pripojená na radič IDE, rozumela príkazom pre SCSI zbernicu, ktoré bude vysielat' napal'ovacia utilita. To práve dosiahneme IDE-SCSI emuláciou.

Podpora tejto emulácie môže byť zahrnutá priamo v jadre alebo ako modul. (Dúfam, že už vieme, čo je to modul a ako sa s ním narába!).

My ešte nevieme upravovať a kompilovať jadro, preto sa tu tým nebudeme teraz zaoberať. Ale ani nepotrebujeme, lebo dnešná rada jadier (2.4 a vyššie) v každej známejšej distribúcii používa systém modulu a je pripravená na prevádzkovanie IDE-SCSI emulácie.

Aktiváciu emulácie dosiahneme zavedením modulu **ide-scsi** do operačnej pamäte počítača. To môžeme vykonať viacerými spôsobmi – buď klasicky pomocou nám už známeho príkazu *modprobe*, alebo to necháme na boot loader, aby tento modul zaviedol hneď pri štarte systému.

V praxi sa používajú dva typy zavádzačov – boot loaderov: staršie **lilo** a modernejší **grub**.

Ale ako nariadime loaderu, aby daný modul zaviedol?

Stačí, ak upravíme jeho príslušný konfiguračný súbor.

Úprava grub.conf

Aby boot loader zaviedol do pamäti modul *ide-scsi*, musíme dopísať čarovnú formulkú **hdX=ide-scsi**, kde **X** je písmeno CD mechaniky. Ak máme pripojenú prepisovačku ako *master* na sekundárnom radiči, za **X** dosadíme písmeno „c“. Potom formula bude **hdc=ide-scsi**.

Na výpise č.1 je obsah reálneho konfiguračného súboru boot loadera *grub.conf*:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You have a /boot partition. This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hda2
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.18-3)
    root (hd0,0)
    kernel /vmlinuz-2.4.18-3 ro root=/dev/hda2 hdc=ide-scsi
    initrd /initrd-2.4.18-3.img
```

Pozrime sa na predposledný riadok:

```
kernel /vmlinuz-2.4.18-3 ro root=/dev/hda2 hdc=ide-scsi
```

Takto sme jadru (kernelu) ako parameter predali informáciu, aby sa natiahol modul *ide-scsi* a že mechanika *hdc* bude takto emulovaná.

Úprava súboru `lilo.conf`

V konfiguračnom súbore boot loadera `lilo` pridáme vyššie uvedenú formulu s pridaním príkazu **append**. Na výpise č.2 je výňatok zo súboru `lilo.conf`:

```
image=/boot/vmlinuz=2.4.18
label= moj_linux
read-only
append="hdc=ide-scsi"
```

Tým, čo používajú `lilo` len pripomeniem, že na rozdiel od `grubu` je pri každej zmene `lilo.conf` nutné vykonať zapísanie zmien príkazom **lilo**!

Poznámka:

V prípade, že máme prepisovačku pripojenú ešte pred inštaláciou systému, inštalačný program sám rozpozná toto zariadenie a sám upraví súbor `grub.conf` alebo `lilo.conf`. Tu vyššie spomínané úpravy budeme musieť vykonať iba vtedy, ak mechaniku pripájame až po inštalácii systému.

Keď vykonáme príslušné zmeny, vykonáme reštart systému (to je ten jeden z tých dvoch razov do roka!). Aby sme zistili, či máme modul naozaj zavedený do pamäti, urobíme čo?no, Novák?neviete, sadnúť, máte za päť!... takže čo? No predsa spustíme príkaz **lsmod**! Ak modul vo výpise uvidíme, je to v poriadku.

Overenie funkčnosti

Aby sme zistili, či je naša mechanika s emuláciou funkčná, použijeme príkaz **cdrecord** s parametrom **scanbus**:

```
[root@rubin grub] # cdrecord -scanbus
```

Na obrazovke monitora by sa malo zobrazíť niečo podobné môjmu výpisu č.3:

```
Cdrecord 1.10 (i686-pc-linux-gnu) Copyright (C) 1995-2001 Jörg Schilling
Using libscg version 'schily-0.5'
scsibus0:
  0,0,0  0) '_NEC      ' 'NR-7500A      ' '1.20' Removable CD-ROM
  0,1,0  1) *
  0,2,0  2) *
  0,3,0  3) *
  0,4,0  4) *
  0,5,0  5) *
  0,6,0  6) *
  0,7,0  7) *
```

Príkaz `cdrecord` rozpoznal v našom Linuxe jedno zariadenie, a to prepisovačku výrobcu NEC, typ NR-7500A. Ak máme v systéme pripojených alebo emulovaných viac SCSI zariadení, príkaz ich vypíše v tom poradí, ako sú zapojené. Zvláštna trojica čísiel popisuje akúsi adresu zariadenia, v našom prípade je to (0,0,0).

Ak teda máme podobný výsledok, zariadenie aj celý systém je pripravený k zapisovaniu našich vlastných cédečiek.

Nabudúce si to všetko nacvičíme na praktickom príklade.

Linux prakticky ako server/12.časť

V minulej časti sme si o „vypaľovaní“ (prosím – nespájať s výpalníctvom!) v linuxovom serveri rozprávali teoreticky. Dnes si to všetko ozrejníme na praktickom príklade.

Praktický príklad

Na rozdiel od oknoidných programov, kde sa vyberú príslušné adresáre a súbory, ktoré chceme zapísať na médium a potom klikneme na zapisovacie tlačidlo, v riadkovom režime to funguje trochu zložitejšie. Ako prvé musíme vytvoriť obraz budúceho cédečka.

Tvorba obrazu

Tvorba obrazu – **image** (čítaj „imidž“) disku je akýsi medzikrok pri tvorbe cédečka. Obraz – **image** je súbor, v ktorom sú schované dáta, ktoré chceme na médium uložiť, vrátane ich formátu, prístupových práv a súborového systému.

Po vytvorení **image** súboru tento súbor prenesieme – zapíšeme – na médium.

Predstavme si, že chceme zazálohovať na médium adresár **/etc/**, v ktorom sa nachádzajú všetky konfiguračné súbory nášho systému.

Najprv vytvoríme obraz disku. Na vytvorenie image súboru slúži príkaz **mkisofs** (**make ISO file system**), ktorý dokáže spracovať dáta tak, že sú vhodné pre súborový systém používaný na CD diskoch.

Jeho syntaktický zápis je

```
mkisofs parametre názov_image_súboru.iso vstupný_adresár/
```

Pozor na záverečnú lomku, tá je povinná!!!

Takže podľa nášho príkladu zadáme príkaz

```
[root@rubin home] # mkisofs -r -o zaloha.iso /etc/
```

Upozornenie!

Tento súbor sa vytvorí do aktuálneho adresára, teda tam, kde je práve nastavený prompt! Ak chceme súbor vytvoriť inde, musíme zadať celú cestu, napr. /mior/napalovanie/zaloha.iso.

Parameter **-r** zapína tzv. Rock Ridge formát (teraz sa tým netrápme), parameter **-o** definuje, že za ním nasleduje meno image súboru. Súbor máva príponu **.iso**.

V prípade, že používame dlhšie mená súborov, ako je 8.3, pridáme ešte parameter **-J** (Joliet). Tým zachováme kompatibilitu dlhých mien tak, ako je známa napr. v MS Windows.

Overenie obrazu

Keďže image súbor nie je bežne čitateľný a my si chceme overiť, čo sa v ktorom image súbore skrýva, máme možnosť primontovať iso súbor ako spätnú slučku do ľubovlného montovacieho bodu.

Nech máme v našom systéme adresár **/mnt/iso**. To tohto adresára primontujeme obraz image súboru takto:

```
[root@rubin home] # mount -o loop zaloha.iso /mnt/iso
```

Ak sa teraz pozrieme do adresára **/mnt/iso**, uvidíme obsah adresára **/etc**, ktorý chceme zapísať na médium.

Po ukončení prezerania takto sprístupneného image súboru ho nesmieme zabudnúť odmontovať, napr.

```
[root@rubin home] # umount /mnt/iso
```

Zápis obrazu na médium

Ak máme vytvorený image súbor a presvedčili sme sa o jeho obsahu, pristúpime k jeho zápisu na médium. Na zápis použijeme opäť príkaz **cdrecord**:

```
[root@rubin home] # cdrecord -v speed=4 dev=0,0,0 -data zaloha.iso
```

Po stlačení klávesu *Enter* nastane odpočítavanie 9 sekúnd, počas ktorého máme možnosť celý proces zapisovania stopnúť a nakoniec prebehne samotný zápis.

Parameter **-v** značí, že na obrazovku sa budú vypisovať aktuálne informácie, čo sa práve so zápisom deje.

Parameter **speed** určuje, akou rýchlosťou sa bude zapisovať. V prípade, že zadáme rýchlosť vyššiu, akou je mechanika v skutočnosti schopná zapisovať, použije sa jej maximálna rýchlosť.

(Niektorá sa teraz začuduje, prečo sa štandardne nepoužíva maximálna rýchlosť. Musíme si uvedomiť, že rýchlosť zápisu nezáleží len od schopnosti mechaniky, ale aj od kvality média. Ak je médium schopné zápisu iba štvornásobnou rýchlosťou a my mechanike určíme, aby zapisovala 12 – násobne, určite médium zničíme!

Dostupná rýchlosť média je zapísaná na obale cédečka, preto sa dobre pozerať, čo kupujeme).

Parameter **dev** je už vyššie spomínaná adresa zariadenia, na ktorom zapisovanie prebieha. Spomeňme si, že túto adresu zistíme pomocou príkazu *cdrecord -scanbus*.

Parameter **-data** stanovuje, že sa jedná o dátové CD, teda počítačové dáta.

zaloha.iso je názov nášho image súboru.

Konfigurácia príkazu *cdrecord*

Priznajme, že vyššie uvedený zápis je trochu zdĺhavý. Parametre, zadávané na príkazovom riadku ako argumenty príkazu *cdrecord* môžeme nadefinovať v konfiguračnom súbore *cdrecord.conf*.

Príklad obsahu tohto súboru je na výpise č.4:

```
#ident @(#)cdrecord.dfl 1.2 00/04/16 Copyr 1998 J. Schilling
#
# This file is /etc/cdrecord.conf
# It contains defaults that are used if no command line option
# or environment is present.
#
# The default device, if not specified elsewhere
#
CDR_DEVICE=NEC

#
# The default speed, if not specified elsewhere
#
CDR_SPEED=4

#
# The default FIFO size if, not specified elsewhere
#
CDR_FIFOSIZE=4m

#
# The following definitions allow abstract device names.
# They are used if the device name does not contain the
# the characters ',', ':', '/' and '@'
#
# drive name    device  speed  fifosize driveropts
#
NEC=            0,0,0   -1     -1      ""
#panasonic=     1,4,0   -1     -1      ""
#plextor=       1,4,0   12     -1      ""
#sanyo=         1,4,0   12     8m      burnproof
#yamaha=        1,5,0   -1     -1      ""
#cdrom=         0,6,0   2      1m      ""
```

Parameter **CDR_DEVICE** definuje meno, pod akým bude mechanika známa v tomto súbore. Hodnotu tohto parametra si môžeme sami zvoliť, napr. *moja_prepiska* a podobne.

V tomto príklade je hodnotou reťazec **NEC** (podľa výrobcu mechaniky).

Druhým parametrom je **CDR_SPEED**. Hodnota tohto parametra udáva rýchlosť, akou budeme na médium zapisovať.

Parameter **CDR_FIFOSIZE** udáva veľkosť bufferu FiFo a nebudeme ju meniť.

Pozor!

Všetky tu zadefinované parametre budú defaultné, ale ak v príkazovom riadku zadáme inú hodnotu, bude platiť tá z príkazového riadku.

Poslednou sekciou je nadefinovanie parametrov mechaniky alebo viacerých mechaník, ak sa v systéme nachádzajú.

Naša mechanika s názvom **NEC** je pripojená na adrese 0,0,0. Čo to znamená už vieme. V stĺpoch *speed* a *fifosize* sa nachádza číslo -1, ktoré značí, že sa použijú parametre, nadefinované o niekoľko riadkov vyššie v tomto súbore.

Predstavme si, že by sme mali pripojenú aj mechaniku, ktorej by sme dali meno **sanyo**. Potom jej parametre z tejto sekcie by boli: adresa 1,4,0, zapisovacia rýchlosť 12 a veľkosť bafra (bufferu) sa rovná 8 MB. Na tomto riadku je ešte v stĺpci *driveropts* hodnota *burnproof*. To značí, že mechanika *sanyo* je schopná používať funkciu *burnproof*.

Čo je to burnproof?

V minulosti, keď boli ešte prvé počiatky zapisovania na CD médiá, bolo nutné zabezpečiť dostatočný tok zapisovaných dát. V prípade, že tento tok dát z určitého dôvodu poklesol (napr. počítač vykonával ešte inú činnosť okrem zápisu), súvislý zápis sa prerušil a médium bolo poškodené a nepoužiteľné.

Preto konštruktéri vymysleli takú vec, že keď poklesne tok dát, laser v mechanike sa vypne, zapisovanie sa preruší a po „dotečení“ dát sa pokračuje na tom mieste, kde sa skončilo a pokračuje sa ďalej. Tejto funkcii sa hovorí **burnproof**.

Keďže viem, že moja mechanika NEC nemá funkciu *burnproof*, nemá tento parameter uvedený v súbore *cdrecord.conf*.

Ako zistíme, či mechanika túto funkciu podporuje? Existuje niekoľko možností:

Buď to zistíme z návodu k mechanike alebo na Internete pohládame nejaký datasheet na stránke výrobcu alebo sa na to rovno spýtame priamo mechaniky.

Ale ako?

Zadáme príkaz:

```
[root@rubin grub] # cdrecord -dev=0,0,0 -driveropts=help -checkdrive
```

Dostaneme výpis č.5:

```
Cdrecord 1.10 (i686-pc-linux-gnu) Copyright (C) 1995-2001 Jörg Schilling
Using libscg version 'schily-0.5'
Device type   : Removable CD-ROM
Version       : 0
Response Format: 1
Vendor_info   : '_NEC   '
Identifikation : 'NR-7500A   '
Revision      : '1.20'
Device seems to be: Generic mmc CD-RW.
Driver options:
None suported for this drive .
```

Nás zaujíma sekcia **Driver options:** (voľby ovládača). Posledná veta hovorí, že táto mechanika nepodporuje žiadne voľby, teda ani *burnproof*.

Keby sme mali pripojenú mechaniku, napr. Sanyo, sekcia *Driver options* by vyzerala takto:

```
Driver options:
burnproof Prepare writer to use Sanyo BURN-Proof technology
noburnproof Disable using Sanyo BURN-Proof technology
```

Podľa výpisu vidíme, že môžeme použiť dva parametre: parameter **burnproof**, ktorý zapne používanie funkcie Burn-Proof alebo parameter **noburnproof**, ktorý túto funkciu v danej mechanike vypne.

Keď máme upravený konfiguračný súbor *cdrecord.conf*, stačí, aby sme zadali trochu kratší príkaz:

```
[root@rubin grub] # cdrecord -v -data zaloha.iso
```

A naše prvé cédečko máme vytvorené. Dobré si ho odložme na pamiatku, pretože odtiaľto to bude pre nás iba rutina.

Zhrnutie

Aby sme si to ujasnili, urobíme si krátke zhrnutie:

- 1) súbory (vrátane podadresárov) na napálenie si pripravíme do niektorého voľného adresára, napr. */home/data/cd1/*
vytvoríme obraz budúceho cédečka, kde použijeme dlhé názvy súborov, ktorý nazveme *zaloha.iso* príkazom

```
[root@rubin home] # mkisofs -r -J -o zaloha.iso /home/data/cd1/
```

Vytvorený obraz môžeme skontrolovať primontovaním do ľubovôlej adresára, napr. */mnt/iso* takto:

```
[root@rubin home] # mount -o loop zaloha.iso /mnt/iso
```

- 2) Obraz po skontrolovaní odmontujeme! (*umount /mnt/iso*)
Po vytvorení obrazu v adresári */home/* vložíme do CD mechaniky prázdne CD médium (často sa na to zabúda!)
- 3) Vytvorený obraz zapíšeme na médium príkazom

```
[root@rubin home] # cdrecord -v -data zaloha.iso
```

- 4) Po zapísaní vyberieme médium z mechaniky, prípadne ho môžeme skontrolovať klasickým namontovaním napr. *mount -t iso9660 /dev/cdrom /mnt/cdrom* (toto je len pre zopakovanie, to už by sme mali vedieť!).
- 5) Pochválime sa svojim priateľom a dáme si kávu...

Praktické rady

Niekedy potrebujeme vypáliť kópiu cédečka. Môžeme to urobiť dvoma spôsobmi:

- Ø priamym kopírovaním z CD na CD bez vytvorenia image súboru
- Ø s vytvorením image súboru

Priame kopírovanie z CD na CD

Aby sme mohli urobiť priame kopírovanie z CD na CD, musíme mať v počítači okrem zapisovacej mechaniky ešte jednu – obyčajnú CD-ROM mechaniku. Nech je táto mechanika známa v systéme ako */dev/cdrom*.

Takže urobíme:

- 1) vložíme zdrojové cédečko do CD-ROM mechaniky
- 2) vložíme cieľové čisté (prázdne) médium do zapisovacej mechaniky
- 3) zadáme príkaz:

```
[root@rubin home] # cdrecord -v -isosize /dev/cdrom
```

- 4) médiá vyberieme

Vytvorenie image z cédečka

Obraz – image súbor z cédečka vytvoríme takto:

- 1) zdrojové CD vložíme do CD-ROM mechaniky a primontujeme
- 2) príkazom:

```
[root@rubin home] # dd if=/dev/cdrom of=/data/iso_cd.iso
```

vytvoríme image súbor s názvom *iso_cd.iso* do adresára */data*.

Takto vytvorený obraz jednoducho vyššie spomínaným spôsobom vypálime.

Poznámka:

Ak bolo zdrojové CD bootovateľné, bude aj jeho kópia bootovateľná.

Vytváranie zvukových stôp

Ak chceme vytvoriť zvukové cédečko, tak urobíme:

- 1) do niektorého adresára si uložíme hudobné súbory typu wav, cdr, au a iné
- 2) zápis vykonáme príkazom

```
[root@rubin home] # cdrecord -v -audio track1.wav track2.au.....
```

alebo jednoduchšie

```
[root@rubin home] # cdrecord -v -audio *.wav
```

Skúška zápisu bez laseru

V prostredí MS Windows sme sa mohli stretnúť so simuláciou zápisu. To je stav, kedy sa vyskúša zápis, ale laser je vypnutý. Takto môžeme otestovať, či pri ozajstnom zápise nedôjde k chybe.

Simulácia je umožnená aj v Linuxe. Stačí, ak použijeme parameter **-dummy**, napr. takto:

```
[root@rubin home] # cdrecord -v -dummy -data zaloha.iso
```

Výmaz CD-RW média

Ak používame CD-RW médiá, a chceme ich vymazať, použijeme jednoduchý parameter **blank=hodnota**. Ak nevieme, aké hodnoty môžeme použiť, zadáme príkaz:

```
[root@rubin home] # cdrecord blank=help
```

Najčastejšie sa používa hodnota **all**, disk alebo **session**, napr.:

```
[root@rubin home] # cdrecord blank=all
```

Je logické, že výmaz musíme urobiť ešte pred zápisom nových dát.

V tejto oblasti existuje ešte mnoho príkladov, ktoré sme tu neprebrali. K niektorým z nich sa ešte dostaneme.

Praktické využitie

Ako je už v texte naznačené, zapisovanie na CD médiá je praktické z dôvodu archivácie dát alebo konfiguračných súborov. V praxi náš linuxový server pravidelne každý druhý deň zálohuje vybrané dáta a adresáre o 4. hodine ráno a ešte o tom pošle hlásenie mailom.

Na to pravidelné zálohovanie, ale nielen na to, sa používa zvláštny program, ktorý sa nazýva **cron**.

Ale o ňom nabadúce.

Linux prakticky ako server/ 13.časť

V minulej časti sme sa naučili pripojiť a ovládať CD-RW mechaniku a naučili sme sa postup, ako na CD-R alebo CD-RW médium preniesť vybrané súbory.

Verím, že mnohí z nás si vytvorili konkrétne skripty na zautomatizovanie činnosti archivácie dát.

Veľa z vás žiada, aby sme sa naučili pod Linuxom sprevádzkovať rôzne zariadenia, napr. zvukovú kartu, wi-fi kartu, grafickú alebo televíznu kartu či pridanie ďalšieho pevného disku. Keďže je to dôležitá téma, budeme sa týmito zariadeniami postupne venovať.

Obecné zásady pri sprevádzkovaní zariadení

Tak ako pod inými operačnými systémami, aj pod Linuxom existujú určité obecné zásady pri tejto činnosti.

Môžeme si ich rozdeliť na:

- Ø fyzická montáž zariadenia
- Ø kompilácia a príslušného ovládača a jeho zavedenie do pamäte
- Ø nastavenie konkrétnych parametrov zariadenia

Tieto obecné postupy sa čiastočne líšia v závislosti od druhu zariadenia, ale podstata je rovnaká.

Najlepšie bude, ak si to ukážeme na konkrétnom príklade.

My si dnes vyberieme sprevádzkovanie wi-fi karty, konkrétne typu *XI-626* od firmy *ZCOMAX* – obr.č1:



Popis karty

Uvedená wi-fi karta patrí medzi najobľúbenejšie zariadenia v bezdrôtových sieťach, používaných pod Linuxom. Je to vďaka výbornej podpore čipsetu danej karty. Je to čipset *Intersil Prism* a s ním veľmi dobre komunikuje ovládač s názvom **hostap**.

Táto karta síce patrí medzi tie drahšie, ale má jednu obrovskú výhodu – dokáže softvérovo pracovať vo viacerých módoch, vrátane prístupového bodu – *access pointu*. Preto stačí jedna takáto karta v linuxovom serveri a zo servera sa stáva prístupový bod. Vtedy nemusíme kupovať drahé externé zariadenie. Zároveň dokáže pod Linuxom (pod MS Windows nie!!!) softvérovo ovládať svoj vysielač výkon. Kto žije v oblasti, kde to je v éteri preplnené, vie, o čom hovorím. Vzájomné rozumné regulovanie výkonu je nielen povinné, ale aj prispeje k vzájomnému nerušeniu okolitých podobných zariadení.

Wi-fi sieť

O podstate wi-fi sietí sme si už rozprávali niekoľkokrát, podrobnejšie v 10.časti seriálu *Linux prakticky ako desktop*, keď sme sprevádzkovali wi-fi kartu v desktope pomocou grafických utilít. My si tieto znalosti rozšírime o tie, ktoré potrebujeme z pohľadu servera.

Fyzická montáž zariadenia

Každé interne pridávané zariadenie do ľubovôlej operačného systému vyžaduje zásah do útrobov počítača. To vyžaduje určitú opatrnosť a dôslednosť.

V prvom rade **nikdy nepridávame žiadne zariadenie za behu počítača!** Nielenže je tu nebezpečenstvo ohrozenia života pri zásahu elektrickým prúdom, ale s veľkou pravdepodobnosťou dôjde k poškodeniu nielen vkladného zariadenia – v našom prípade wi-fi karty, ale aj k poškodeniu základnej dosky počítača. Pri montáži položíme dôraz na riadne zasunutie nožového konektora (to sú tie pozlátené kontakty) do slotu PCI na základnej doske počítača (niektoré karty majú tendenciu sa trochu vzpierať, menšie násilie nie je na škodu veci).

Po zasadení prichytíme skrutkou plechový držiak o skrinku počítača, čím sa vyhneme samovoľnému vysunutiu konektora zo slotu.

Nakoniec naskrutkujeme dodávanú anténu.

Pozor!

Jedná sa o rádiové (vysielacie) zariadenie, takže opomenutie namontovania antény môže viesť k poškodeniu koncového stupňa wi-fi karty!

Kompilácia príslušného ovládača

Ak sa príslušný ovládač nenachádza na doprovodnej diskete alebo cédečku, musíme si ho stiahnuť z Internetu. Je dobré si na Internete skontrolovať, či sa k nášmu zariadeniu nenachádza ovládač v novšej verzii, ako je na diskete či CD disku. Prax potvrdzuje, že to tak spravidla naozaj býva. Novší ovládač odstraňuje nedostatky predchádzajúcich verzií (alebo aspoň by mal, nie?).

Vieme, že vhodný ovládač pre našu wi-fi kartu sa volá **hostap** a ten si stiahneme z adresy

<http://hostap.epitest.fi/releases/>

alebo vylepšený od p.Šimandla z Čech na adrese

http://www.simandl.cz/stranky/linux/xi626/soubory/hostap_cvs_030220_200_01.tar.gz

Aby sme mohli ovládač *hostap* používať, potrebujeme mať v systéme ešte tieto veci:

- Ø jadro Linuxu s podporou *Wireless Extensions*
- Ø konfiguračné nástroje z balíku *Wireless Tools*

Jadro s podporou *Wireless Extensions*

Ovládanie wi-fi kariet je v jadre riešené pomocou definovania súboru parametrov, ktoré sa nazývajú *Wireless Extensions*. V súčasnej dobe má už každé jadro podporu *Wireless Extensions* v sebe zapnutú a zkompilovanú. (v prípade, že by tak nebolo, musíme jadro opatchovať (zaplátať) a znovu skompilovať. Predpokladajme, že používame jadro verzie 2.4.x, kde sa nachádzajú WE vo verzii 15, preto sa nebudeme teraz zaoberať, ako WE v jadre aktivovať).

Konfiguračné nástroje z balíku *Wireless Tools*

Wireless Tools sú nástroje na konfigurovanie parametrov wi-fi kariet a wi-fi spojenia. *Wireless Tools* nájdeme na adrese <http://pcmcia-cs.sourceforge.net/ftp/contrib/> vo verzii 26.

Je však veľmi pravdepodobné, že sa v našej distribúcii už nachádzajú spolu s *Wireless Extensions*.

Či je to skutočne tak, to si overíme príkazom:

```
[root@rubin net] # iwconfig -version
```

Na obrazovke sa objaví výpis č.2:

```
iwconfig Version 26
Compatible with Wireless Extension v16 or earlier,
Currently compiled with Wireless Extension v15.
```

Z výpisu sme zistili, že v systéme máme nainštalované *Wireless Tools* verzie 26 a *Wireless Extensions* verzie 15.

Teraz pristúpime ku kompilácii ovládača *hostap*.

Kompilácia ovládača

Stiahnutý ovládač, napr. *hostap_cvs_030220_200_01.tar.gz* uložíme do pomocného adresára, nech je to napr. */install*.

Skomprimovaný ovládač rozbalíme pomocou príkazu *tar* alebo využijeme dobrého pomocníka *Midnight Commander*.

V uvedenom adresári vyhladáme súbor *Makefile*. Vo vhodnom editore (napr. *vi* alebo zase použijeme *mc*) editujeme tento súbor.

Vyhladáme riadok `KERNEL_PATH`.

Upravíme cestu ku zdrojovým kódom jadra nášho systému, napr.:

```
KERNEL_PATH=/usr/src/linux-2.4
```

Súbor po editovaní uložíme.

V danom adresári spustíme príkaz:

```
[root@rubin install] # make pci
```

Tento príkaz zkompiluje zdrojové kódy a vytvorí hlavný súbor *hostap_pci.o* a pomocné súbory *hostap.o*, *hostap_crypt.o* a *hostap_crypt_wep.o*.

Potom zadáme príkaz

```
[root@rubin net] # make install_pci
```

Tento príkaz uloží uvedené súbory do príslušného adresára v adresárovej štruktúre, kde má operačný systém uložené ovládače jednotlivých zariadení. V prípade distribúcie *Fedora Core 1* je to adresár `/lib/modules/2.4-22-1.2115.nptl/net/`.

Zavedenie ovládača do pamäte

My už vieme, čo je ovládač a že ho môžeme zaviesť do pamäte príkazom

```
[root@rubin net] # modprobe hostap_pci
```

Ak príkaz nevyhlásil žiadnu chybu, môžeme príkazom

```
[root@rubin net] # lsmod|grep hostap
```

overiť, že sa príslušné ovládače nachádzajú v operačnej pamäti – výpis č.3:

hostap_pci	42076	0	(unused)
hostap	71652	0	[hostap_pci]
hostap_crypt	2768	0	[hostap]

Ak chceme moduly z pamäte odstrániť, použijeme príkaz

```
[root@rubin net] # rmmod -r hostap_pci
```

Poznámka:

Mnohí mi v mejloch právom vytýkate, že sa pri niektorých príkazoch pozastavujem a vysvetľujem ich opakovane dopodrobna, aj keď už raz boli vysvetlené. Druhí mi zase píšete, čo je to za príkaz, napríklad lsmod alebo grep. Upozorňujem, že je to látka, ktorú sme už preberali a ja sa nemôžem donekonečna zaoberať vysvetľovaním už raz vysvetlených vecí. Preto vás dôrazne a vo vašom záujme žiadam, aby ste sa učili priebežne a ak niečo pozabudnete, aby ste si to samostatne doštudovali. Ani v škole učitelia nestrpia neznalosti preberanej látky [preto verím, že viete, čo lsmod|grep znamená!!!]

Pri zavedení ovládača do pamäte sa zariadeniu prideli názov **wlanX**, teda ak máme iba jednu kartu, tak to bude **wlan0**.

Tááák, ovládač by sme v pamäti mali, teraz nastavíme parametre zariadenia.

Nastavenie parametrov zariadenia

Na nastavenie konkrétnych parametrov wi-fi karty slúžia práve *Wireless Extensions* a *Wireless Tools* a nám už dobre známy príkaz **ifconfig**.

Čo vlastne potrebujeme nastavovať?

Musíme si uvedomiť, že wi-fi karta sa správa ako klasická sieťová karta, len namiesto drôtov používa rádiový prenos. Preto potrebuje nastaviť ako sieťové, tak aj rádiové parametre.

Medzi sieťové parametre patrí:

- Ø názov zariadenia
- Ø IP adresa a maska

Medzi základné rádiové parametre patrí:

- Ø **mode** - mód, v ktorom zariadenie pracuje
- Ø **ESSID** – označenie siete
- Ø **rate** - rýchlosť prenosu
- Ø **channel** - prenosový kanál
- Ø **key** - šifrovanie a jeho kľúč
- Ø **txpower** - vysielací výkon (ak jeho reguláciu karta podporuje)

Sieťové parametre nastavujeme príkazom *ifconfig*, rádiové parametre budeme nastavovať utilitou z *Wireless Tools*, ktorá je veľmi podobná príkazu *ifconfig*, ale slúži pre wi-fi zariadenia. Nazýva sa ***iwconfig***.

Príkaz *iwconfig* má mnoho parametrov, ktorými sa dané zariadenie konfiguruje. Nebudeme ich teraz preberať všetky, ukážeme si tie najdôležitejšie v našom praktickom príklade. Ostatné si doštudujte z manuálových stránok.

Praktický príklad

Predpokladajme, že máme v našom linuxovom serveri namontovanú spomínanú wi-fi kartu.

Najprv zavedieme ovládač do pamäte:

```
[root@rubin net] # modprobe hostap_pci
```

Vieme, že dostane názov *wlan0*. Aby sme ho mohli konfigurovať, najprv musíme toto zariadenie „naštartovať“:

```
[root@rubin net] # ifconfig wlan0 up
```

My chceme, aby táto karta fungovala ako prístupový bod (access point), takže mód zariadenia bude *master*:

```
[root@rubin net] # iwconfig wlan0 mode master
```

IP adresa tejto karty bude 192.168.100.1 a maska 255.255.255.0:

```
[root@rubin net] # ifconfig wlan0 inet 192.168.100.1 netmask 255.255.255.0
```

Teraz nastavíme označenie siete (ESSID), ktorým sa bude naša sieť prezentovať v eteri:

```
[root@rubin net] # iwconfig wlan0 essid "skuska"
```

A nakoniec nastavíme vysielací výkon:

```
[root@rubin net] # ifconfig wlan0 txpower 1
```

Ak ostatné parametre nenastavíme, použijú sa defaultné nastavenia.

Či je karta naozaj riadne nastavená podľa našich požiadaviek, to si overíme vyššie spomenutými príkazmi.

Rádiové parametre overíme príkazom

```
[root@rubin net] # iwconfig wlan0
```

Na obrazovke počítača by sa mal nachádzať text podobný tomu na výpise č.4:

wlan0	IEEE 802.11b	ESSID:"skuska"	
	Mode:Master	Frequency:2.422GHz	Access Point: 00:60:B3:6D:8D:89
	Bit Rate:11Mb/s	Tx-Power=1 dBm	Sensitivity=1/3
	Retry min limit:8	RTS thr:off	Fragment thr:off
	Encryption key:off		

```
Power Management:off
Link Quality:0   Signal level:0   Noise level:0
Rx invalid nwid:0   Rx invalid crypt:0   Rx invalid frag:0
Tx excessive retries:0   Invalid misc:2   Missed beacon:0
```

Sieťové parametre overíme príkazom:

```
[root@rubin net] # ifconfig wlan0
```

Na obrazovke počítača by sa mal nachádzať text podobný tomu na výpise č.5:

```
wlan0      Link encap:Ethernet  HWaddr 00:60:B3:6D:8D:89
            inet addr:192.168.100.1  Bcast:192.168.100.255  Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
            Interrupt:10 Memory:e09f2000-e09f3000
```

Štartovacie skripty

My už sme skúsení serveristi a vieme, že postupné zadávanie týchto príkazov priamo z konzoly nie je to pravé orechové. Vhodnejšie je urobiť si nejaký skript, v ktorom budú tieto príkazy sústredené na jednom mieste. Tento skript potom môžeme spúšťať jednoduchým volaním z konzoly alebo ho spustíme z niektorého zo štartovacích skriptov.

Ale ani to nie je úplne ono!

Spomeňme si na tému, kde sme si vysvetľovali, ako Linux štartuje. Povedali sme si niečo o úrovniach behu a o príslušných skriptoch v nich, ovládání rôznych služieb a podobne.

Ved' sprevádzkovanie wi-fi na našom serveri je v podstate služba, nie? Tak prečo to celé neautomatizovať?

Na výpise č. 6 je obsah vzorového skriptu služby z názvom **wifi**, ktorý pomocou príkazu *chkconfig* zavedieme do príslušných adresárov:

```
#!/bin/bash
#
# chkconfig: 35 13 86
# description: Spustanie wifi na XI-626 wlan0
#
# mior 02/2004

# source function library
. /etc/rc.d/init.d/functions

RETVAL=0

case "$1" in
    start)
        echo -n "Startovanie wifi: "
        modprobe hostap_pci
        ifconfig wlan0 up
        iwconfig wlan0 mode master
        ifconfig wlan0 inet 192.168.100.1 netmask 255.255.255.0
        iwconfig wlan0 essid "skuska"
        iwconfig wlan0 txpower 1
        RETVAL=$?
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/wifi
        echo
        ;;
    stop)
        echo -n "Vypinanie wifi: "
```

```
    ifconfig wlan0 down
    rmmmod -r hostap_pci
    RETVAL=$?
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/wifi
    echo
    ;;
restart)
    $0 stop
    $0 start
    RETVAL=$?
    ;;
*)
    echo "Použitie: wifi {start|stop|restart}"
    exit 1
esac

exit $RETVAL
```

Po nainštalovaní môžeme túto službu ovládať pomocou príkazov

```
[root@rubin net] # service wifi start|stop|restart
```

Zároveň je táto služba automaticky aktivovaná pri štarte systému (doporučujem vrátiť sa k častiam o službách a manipulácii s nimi!).

Iný čipset – iný ovládač

Vyššie uvedený spomínaný príklad je naozaj funkčný. Úspešne ho používam denno-denne. Ovládač *hostap* môžeme použiť aj pre wi-fi kartu od iného výrobcu. Podmienkou je rovnaký čipset *Intersil Prism*. V prípade, že budeme chcieť použiť inú wi-fi kartu s iným čipsetom, nemôžeme použiť ovládač *hostap*.

V poslednej dobe sa objavili veľmi lacné wi-fi karty s čipsetom RTL8180. Stačí si stiahnuť zdrojové kódy z Internetu, prečítať súbory README a INSTALL a dodržať predpísaný postup.

V nich sa budú nachádzať popisy na preklad ovládača zo zdrojových kódov. Ostatné tu popísané časti, ako je nastavovanie parametrov daného zariadenia, môžeme využiť aj pri iných čipsetoch. Tak nech nám to v éteri pekne sviští!

Linux prakticky ako server aj ako desktop / 14.časť

Ešte než začneme...

Už to tak občas býva aj v živote. Dvaja rôzni ľudia existujú vedľa seba, kľudne, paralelne. Niekedy sa ich cesty nakrátko stretnú, pobudnú spolu a potom sa zase rozídu. Ani v linuxovej brandži to nebýva inak. A preto sa dnes desktopisti aj serveristi stretnú u jedného článku. Prečo?

Lebo téma dnešného čísla je pre obidve skupiny spoločná. Nájdu sa v nej jedni aj druhí, pretože ako serveristi, tak aj desktopisti aspoň raz budú potrebovať zostaviť nové jadro svojho systému.

A tomu sa dnes budeme venovať.

Otvorenosť zdrojových kódov

Asi najsilnejšou stránkou Linuxu je otvorenosť jeho zdrojových kódov. To znamená, že tieto kódy sú dostupné komukoľvek a v rámci zásad verejnej licencie GNU môže ktokoľvek robiť do nich zásahy a takto pozmenené zdrojové kódy ďalej šíriť.

Ako príklad použijeme program *mc* (Midnight Commander). Predstavme si, že sme celkom zdatní programátori a chceme do tohto výborného programu pridať nejakú novú črtu – slangovo povedané fičúrku. Zoberieme zdrojové kódy programu *mc*, urobíme príslušnú zmenu a program prekompilujeme. Takto pozmenený program, vrátane pozmenených zdrojových kódov ďalej rozšírime medzi linuxovú verejnosť. Ako sa nová funkcia ujme, to overí čas a skutočnosť, či ju tam niekto nechá, prípadne ešte vylepší alebo ju ako chybnú a nepotrebnú vyradí. Je to jeho slobodná vôľa a o tom celý Linux a slobodný softvér je!

Tak ako je možné pozmeniť (samozrejme že k lepšiemu) akýkoľvek program šírený pod licenciou GNU (pozor! Nie všetky programy pre Linux sú free!), tak môžeme vylepšovať aj samotné jadro Linuxu.

Keďže jadro je najdôležitejšia súčasť operačného systému, existujú určité pravidlá, ako zavádzať nové zmeny do jadra. Všetky zmeny, týkajúce sa jadra, sa hlásia a poskytujú určitej skupine programátorov, venujúcich sa jeho vývoju. Daná zmena prejde mnohými testami a overovaním a na konci je vyjadrenie vývojárov, či sa daná zmena objaví v novej verzii jadra alebo nie. Vývojová skupina jadra (*kernel maintainers*) sa skladá z vývojárov mnohých krajín, ktorým „predsedá“ sám Linus – otec Linuxu. Za všetkých spomeňme aspoň Alana Coxa z Veľkej Británie a Alexeja Kuznecova z Moskvy.

Výhodou tohto postupu je predovšetkým rýchlosť. Všetky zmeny, ale aj prípadné chyby sa vďaka Internetu oznamujú veľmi rýchlo a tak rýchlo vznikajú prípadné opravy.

U iných, spravidla komerčných a hlavne uzavretých (proprietárnych) operačných systémov sme zvyknutí aj dlhé mesiace čakať, až príslušná spoločnosť chybu odstráni a vydá nový operačný systém s číslom XY, ktorý musíme od podlahy celý nainštalovať znova.

V Linuxe môžeme prípadné chyby jadra okamžite po zverejnení opravy, čo trvá niekedy iba hodiny, riešiť „záplatami“ alebo stiahnutím novej verzie jadra s následným zostavením bez nutnosti reinstalácie celého systému!

Dôvody na zostavenie jadra

Okrem vyššie spomínaného prípadu opravy chyby jadra alebo vydania novej verzie jadra môžu nastať aj iné prípady, kedy je potrebné jadro znovu zostaviť.

Býva to vtedy, keď chceme používať určitú systémovú utilitu a tá vyžaduje, aby bola v jadre zapnutá podpora pre danú funkciu. Veľmi často sa to týka správy sieťových vecí, ako je napríklad monitorovanie siete, pridelovanie prenosového pásma jednotlivým používateľom, riešenie firewallu a podobne.

Druhým prípadom býva situácia, keď do počítača pridávame nový hardvér, napríklad magnetopáskovú jednotku a potrebujeme v jadre aktivovať pre ňu podporu.

No a tretím príkladom môže byť skutočnosť, že chceme pod Linuxom sprevádzkovať wi-fi sieť a nemali sme doteraz defaultne túto podporu zapnutú.

Najčastejším prípadom však býva skutočnosť, že chceme jadro optimalizovať vzhľadom na náš hardvér, hlavne na typ procesoru a čipovú sadu základnej dosky počítača.

Poznámka:

Vo veľkých distribúciách sa spravidla nachádzajú už priemerne nastavené a „vyladené“ jadrá k predpokladanému účelu, preto nebýva nutné nastavenie jadra zmeniť a znovu zostavovať.

Ale ak v takejto distribúcii chceme vymeniť jadro za novšie, napríklad z dôvodu, že nové jadro podporuje SATA disky, nastaveniu a rekompilácii sa nevyhneme!

Ešte predtým, ako sa pustíme do samotného zostavovania jadra, musíme si o jadre povedať niečo viac.

Jadro

Jadro – anglicky *kernel* – je najzákladnejší program operačného systému Linux. Jeho úlohou je tvoriť rozhranie medzi hardvérom počítača a ostatnými softvérovými aplikáciami.

Jadro má veľa funkcií. Ako sme si povedali, základnou funkciou je sprostredkovanie komunikácie medzi technickými prostriedkami počítača a vytváranie funkčného prostredia pre úspešný beh ostatných aplikácií. Toto prostredie musí zabezpečiť činnosť siete, prístup k diskom, disketovým jednotkám a iným mechanikám, vytvorenie a beh virtuálnej pamäte a súbežné – paralelné spracovávanie úloh, ktorému hovoríme *multitasking*. Jadro sa musí správať tak, aby ten istý program, tá istá aplikácia alebo utilita fungovala na počítači s rôznymi typmi procesorov.

Utility a programy, tvoriace systémové príkazy a „obaľujúce“ jadro tvoria operačný systém. Príkladom môžu byť utility na administráciu používateľov, formátovanie a obsluhu diskov a disketových mechaník a podobne. Aplikácie, spolu s jadrom a systémovými utilitami operačného systému tvoria distribúciu. Príkladom aplikácií sú rôzne editory, komunikačné programy ale aj grafická nadstavba.

Z toho vyplýva, že v každej slušnej linuxovej distribúcii je jednotné linuxové jadro, len možno trochu prispôbené danej distribúcii. To značí, že program alebo aplikácia z ľubovolnej inej distribúcie by mali fungovať aj v inej distribúcii. Hovorím „by mali“, lebo nie vždy je to pravda. To záleží, ako je program alebo aplikácia tesne „zviazaná“ s materskou distribúciou – teda či používa určitú knižnicu, ktorá sa v inej distribúcii nenachádza.

Skutočnosť je taká, že skoro každý program sa dá „prekonvertovať“ pre druhú distribúciu. Vyžaduje si to však určité skúsenosti a hlavne podrobnejšie znalosti. (Nebojte sa, pomaly ale isto k tomu spejeme).

Pre zaujímavosť treba povedať, že dnešné jadro Linuxu obsahuje 2,8 milióna riadkov zdrojového kódu oproti 9000 riadkom Unixu od firmy Bell Labs z roku 1976 (len tak na okraj, nechcete si ich prečítať? ;-)).

Zdrojové kódy jadra

Nech už je z akéhokoľvek dôvodu nutné použiť nové jadro, musíme najprv získať jeho zdrojové kódy.

Zdrojové kódy jadra nájdeme na Internete na adrese www.kernel.org alebo na slovenskom zrkadle www.sk.kernel.org/pub/linux/kernel/. Tu nájdeme všetky verzie jadra, od historických až po tie najnovšie. Príkladom zdrojového kódu jadra je súbor *linux-2.6.8.tar.gz* o veľkosti 34 MB.

Ak používame niektorú z veľkých distribúcií, je lepšie hľadať kódy nového jadra na stránkach tvorca distribúcie. To z dôvodu, že „distrované“ jadro môže byť trochu upravené a šité pre danú distribúciu a máme záruky, že s ním budú všetky balíky z distribúcie pracovať spoľahlivo. Distribučné jadrá sú spakované balíčkovacím jednotlivých distribúcií, napr. rpm a podobne.

Musíme si uvedomiť, že jadrá s nepárnym druhým číslom v názve (napr. 2.5.3) sú vývojové rady a jadrá s párnym druhým číslom (2.4.x, 2.6.x) sú stabilné rady. Čísla jadier majú ešte dopĺňujúce čísla (a niekedy aj písmená), čím sa odlišujú určité drobné opravy chýb, ktoré nemajú fatálny následok. Príkladom takého číslovania je jadro 2.4.22-1.2115nptl z distribúcie Fedora Core 1.

Veľmi často je možné nájsť zdrojové kódy jadra na rôznych cédečkách s linuxovou tematikou, v rôznych počítačových časopisoch a podobne. No a istotne na inštaláčnom CD každej distribúcie!

Patchovanie jadra

Nie vždy je kvôli malej chybe nutné sťahovať nové opravené jadro. Na opravu chyby môžeme použiť príslušnú záplatu – patch (čítaj peč).

Pre bližšiu názornosť si môžeme záplaty pripodobniť service packom iného známeho operačného systému.

Zatiaľ čo záplatou – patchom sa opravujú zdrojové kódy jadra, service packy opravujú binárne kódy operačného systému.

Pozor!

Nesmieme si pliesť zdrojové a binárne kódy jadra! Zdrojové kódy sú textové súbory, napísané v programovacom jazyku (spravidla v jazyku C), ktorých obsah môžeme dostupnými programami čítať

a editovať. Binárne kódy vzniknú po kompilácii zdrojových kódov príslušným prekladačom (spravidla gcc) a nie je možné ich bežne čítať ani modifikovať.

(To platí všeobecne – až na výnimky - pre každý program, nielen pre jadro).

Preto pri študovaní dnešnej lekcie si musíme uvedomiť, kedy hovoríme o zdrojových kódach a kedy o binárnych kódach!

Zostavovanie jadra

Samotné zostavovanie jadra sa skladá z viacerých úkonov:

- Ø rozbalenie zdrojového kódu jadra (a prípadné opatchovanie)
- Ø konfigurácia jadra
- Ø preklad (kompilácia) jadra
- Ø inštalácia jadra
- Ø inštalácia modulov

Tieto úkony vyžadujú presnú postupnosť a pre to si ich teraz prejdeme *step-by-step*, čiže po našsky krok za krokom. Ako vzor použijeme distribúciu Fedora Core1, ale postupy platia aj pre ostatné distribúcie ako je Red Hat, Mandrake, SUSE a ostatné.

Rozbalenie zdrojového kódu jadra

Ak nechceme použiť nové jadro, len chceme v súčasnom jadre povoliť určité služby jadra, alebo chceme len jadro opatchovať, tento odstavec preskočíme.

Vieme, že zdrojový kód jadra sa nachádza v určitej adresárovej štruktúre, ktorej sa niekedy hovorí aj strom.

V distribúcii FC1 je to adresár `/usr/src/linux-2.4/`. Všimnime si, že je to v skutočnosti linka na adresár `/usr/src/linux-2.4.22-1.2115nptl`.

Aby sme neprišli o to, čo už máme funkčné, doporučujem uvedený adresár prekopírovať na iné miesto alebo ho stačí premenovať, napr. `/usr/src/linux.old/`.

Novo získané jadro, napr. súbor `linux-2.4.27.tar.gz`, uložíme do adresára `/usr/src/`.

Teraz prichádza to najdôležitejšie – rozbalenie jadra.

Ak máme jadro ztarované, použijeme príkaz

```
[root@doma src] # tar -xzf linux-2.4.27.tar.gz
```

Zdrojové kódy jadra sa rozbalia do adresára `/usr/src/linux-2.4.27/`.

V prípade, že sme stiahli súbor s príponou `.bz2`, použijeme príkaz

```
[root@doma src] # tar -xzf linux-2.4.27.tar.bz2 --use-compress-program bzip2
```

Rozbalenie môže trvať aj niekoľko minút v závislosti na rýchlosti počítača, predsa sa len jedná o skoro 3 milióny riadkov!

V prípade, že získame jadro od vydavateľa distribúcie s príponou `.rpm`, použijeme príkaz

```
[root@doma src] # rpm -i kernel-headers*.rpm  
[root@doma src] # rpm -i kernel-source*.rpm
```

Keďže mnohé utility a programy sa odvolávajú na adresár `/usr/src/linux-2.4/`, musíme zároveň upraviť linku na tento adresár. Buď použijeme `mc` alebo z riadku zadáme príkaz

```
[root@doma src] # ln -s linux-2.4.27 linux-2.4
```

(Nesmieme zabudnúť predtým predchádzajúcu linku zrušiť!))!!

Opatchovanie

Ak chceme stávajúce jadro opraviť, použijeme príslušnú záplatu. Záplata má meno podobné číslu jadra, napr. `patch-2.4.19.gz` vytvorí zo starého jadra 2.4.18 jadro nové 2.4.19.

Patch rozbalíme a aplikujeme príkazom

```
[root@doma src] # gunzip -c patch-2.4.19.gz|patch -p1
```

Zdrojové kódy súčasného jadra sú opravené, môžeme prísť ku konfigurácii budúceho jadra.

Konfigurácia jadra

Práve konfigurácia jadra je najdôležitejší bod celého zostavovania. Vďaka tejto filozofii môžeme jadro systému prispôbiť k obrazu svojmu. Môžeme rozhodnúť o dôležitých súčiastiach jadra. Ak napríklad nemáme v počítači žiadne SCSI zariadenie, tak podporu týchto zariadení v jadre vypneme, čím ušetríme drahocenné miesto v operačnej pamäti. Dobrým „vyladením“ konfigurácie jadra získame jadro, ktoré bude dostatočne výkonné aj na starších počítačoch.

Ešte predtým, ako prísť ku samotnej konfigurácii, je dobré si zistiť, aké zariadenia má naše nové jadro podporovať. To zistíme pohľadom do imaginárneho súboru */proc/pci* alebo jeho vylisťovaním príkazom

```
[root@doma src] # cat /proc/pci
```

Uvidíme základný zoznam hardvéru a začneme konfigurovať zdrojové kódy jadra.

Samotnú konfiguráciu môžeme vykonať tromi spôsobmi:

- Ø v príkazovom riadku
- Ø pomocou menu v príkazovom riadku
- Ø pomocou menu v grafickom prostredí

Konfiguráciu v príkazovom riadku spustíme príkazom **make config**, pomocou menu v príkazovom riadku **make menuconfig** a v grafickom prostredí pomocou príkazu **make xconfig**. Tieto príkazy musíme spúšťať v adresári zdrojových kódov jadra, teda v */usr/src/linux-2.4/*.

Aj keď všetky tri možnosti dosiahnu ten istý výsledok, najpríjemnejší z nich je posledný.

V prípade, že na danom počítači nemáme nainštalované grafické prostredie, použijeme druhú možnosť.

Budeme postupovať takto:

- Ø prejdeme do daného adresára */usr/src/linux-2.4/* príkazom

```
[root@doma src] # cd /usr/src/linux-2.4
```

- Ø zadáme príkaz

```
[root@doma linux-2.4] # make menuconfig
```

Na obr.č.1 vidíme základné menu konfiguračného nástroja **menuconfig**:



Ak máme možnosť použiť grafické prostredie, postupujeme takto:

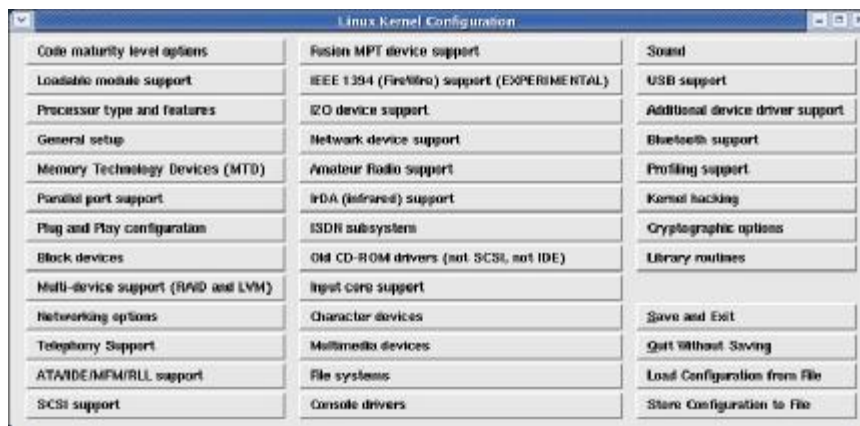
- Ø spustíme grafické prostredie príkazom **startx** s právami roota
- Ø v grafickom prostredí spustíme ľubovoľný terminál (objaví sa klasický shell)
- Ø prejdeme do daného adresára */usr/src/linux-2.4/* príkazom

```
[root@doma src] # cd /usr/src/linux-2.4
```

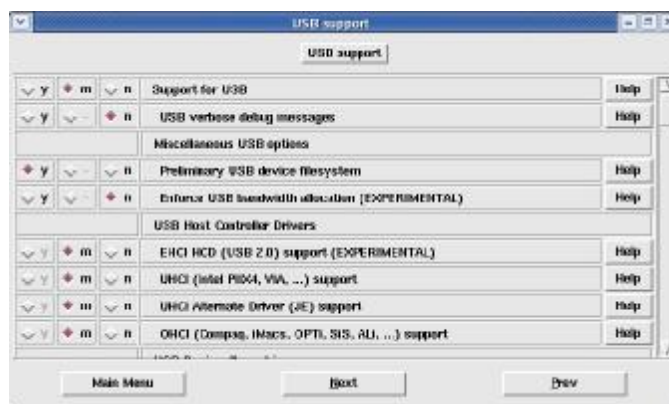
Ø zadáme príkaz

```
[root@doma linux-2.4] # make xconfig
```

Na ploche sa objaví okno konfiguračného nástroja – obr.č.2:



Tento konfiguračný nástroj nám pomôže pri vytváraní skriptov, potrebných pri preklade programu. Ako vidíme na obr.č.3, utilita sa skladá z niekoľkých položiek. Kliknutím na jednotlivé položky sa dostaneme do nižšej úrovne. Kliknime napríklad na položku **USB support** – obr.č.3:



Všimnime si rozdelenie okna **USB support**. Na ľavej strane sú tri stĺpce **y**, **m** a **n**, v strede je názov jednotlivej voľby a vpravo je help - nápoved'. Kliknutím na kosoštvorec niektorého z prvých troch stĺpcov zvolíme danú voľbu, čo sa prejaví jeho sfarbením na červeno. Stĺpec **y** definuje, že daná podpoložka bude povolená a bude neoddeliteľnou súčasťou jadra. Stĺpec **n** definuje, že podpoložka nebude aktívna ani nebude súčasťou jadra. Tým dosiahneme zmenšenie jadra a úsporu pamäte. Stĺpec **m** definuje, že sa má daná voľba preložiť ako modul. My už vieme, čo to moduly sú, aké sú ich výhody a ako sa s nimi narába. Zopakujme si, že táto možnosť je výhodná, pretože nám umožňuje modul aktivovať alebo deaktivovať na základe našich potrieb aj po kompilácii jadra systému. Pozor, nie všetky voľby jadra je možné kompilovať ako modul, vtedy v druhom stĺpci nie je písmeno **m**, ale znamienko *mínus* a táto voľba je neaktívna!

Napríklad jadro musí vedieť prístupovať na pevný disk, prechádzať súborový systém a potom môže žiadané moduly spúšťať. Takže podpora prístupu k pevným diskom nemôže byť ako modul, ale musí byť pevnou súčasťou jadra!

Prechádzať všetky položky konfiguračného nástroja je teraz zbytočné. Ak nastane prípad, že chceme použiť niektorú sieťovú utilitu, v dokumentácii k danej utilite sa bude istote nachádzať pokyn, ktoré voľby konfiguračného nástroja máme zapnúť!

Po ukončení konfigurácie klikneme na položku **Save and Exit** v hlavnom menu.

Tým sa vytvorí v adresári `/usr/src/linux-2.4/` súbor `.config`. V ňom sa nachádzajú uložené všetky vykonané nastavenia. (načo sa tento súbor použije si neskôr ukážeme).

Po vykonaní konfigurácie môžeme ešte vykonať označenie nového jadra.

V adresári `/usr/src/linux-2.4/` vyhľadáme súbor *MakeFile*. Otvoríme ho vhodným editorom a prejdeme na riadok **EXTRAVERSION** = . Tu doplníme ľubovoľný text, ktorým označíme naše nové jadro, napríklad **mojejadro**. Po editovaní súbor uložíme.

Preklad jadra

V globále je preklad jadra najjednoduchší, ale zato časovo náročný. Preklad jadra, ktorému sa odborné hovorí kompilácia, sa skladá z niekoľkých fáz:

- Ø Vytvorenie závislostí
- Ø Mazanie starých a nepotrebných súborov
- Ø Vytvorenie obrazu jadra
- Ø Vytvorenie a inštalácia modulov

Vytvorenie závislostí

Aby systém vedel, ktoré súbory je nutné preložiť a ktoré je možné ignorovať, vytvoríme strom závislostí. Použijeme príkaz (všetko v adresári `/usr/src/linux-2.4/`):

```
[root@doma linux-2.4] # make dep
```

Mazanie

Na zmazanie starých a nepotrebných súborov použijeme príkaz

```
[root@doma linux-2.4] # make clean
```

Tento príkaz sa doporučuje použiť aj vtedy, ak ideme zostavovať úplne nové jadro.

Vytvorenie obrazu jadra

Z vytvoreného konfiguračného súboru, vytvorených závislostí a „vyčisteného stola“ spustíme skutočný preklad zdrojových súborov. Tých zdrojových súborov je neúrekom, stačí sa pozrieť do adresára `/usr/src/linux-2.4/`. Kompiláciu spustíme príkazom

```
[root@doma linux-2.4] # make bzImage
```

Po preklade vznikne iba jeden veľký hlavný súbor, ktorý obsahuje obraz nového jadra a jeden pomocný malý súbor.

Doporučujem preklad robiť pred obedom, lebo aj na dobrých strojoch trvá dosť dlho. No a keď sa vrátíme s obeda, už by to mohlo byť hotové.

(Keď som to robil prvýkrát v živote na Pentiu 90 a RedHat 6.2, trvalo to celú noc...)

Vytvorenie modulov

Ak sme vytvárali nové jadro, musíme prekladom vytvoriť aj moduly, ktoré sme v jednotlivých položkách konfiguračného nástroja nadefinovali. To dosiahneme príkazom

```
[root@doma linux-2.4] # make modules
```

Inštalácia modulov

Po preložení modulov je potrebné moduly nainštalovať do toho správneho adresára a upraviť určité skripty, aby jadro pri bootovaní moduly správne zaviedlo do pamäti. Aby sme to nemuseli robiť ručne, použijeme na to krásny príkaz

```
[root@doma linux-2.4] # make modules_install
```

Tento príkaz nainštaluje moduly do adresára `/lib/modules/číslo_jadra`, teda v našom prípade `/lib/modules/2.4.22-1.2115.nptl/`.

Inštalácia jadra

Jadro, teda jeho obraz sa po preklade nachádza v adresári `/usr/src/linux-2.4/arch/i386/boot/` v súbore *bzImage*. Najprv prepokopírujeme tento súbor do adresára `/boot/` a premenujeme ho súbor *vmlinuz-x.y.z*

```
[root@doma linux-2.4] # cp /usr/src/linux-2.4/arch/i386/boot/bzImage /boot/vmlinuz-2.4.22-mojejadro
```

Keď máme nové jadro hotové, zostáva nám len upraviť príslušný zavádzač. Jeho nastavenie závisí od typu použitého bootloadera, teda či používame GRUB alebo LILO.

Úprava bootloadera GRUB

Prejdeme do adresára `/boot/grub/` a editujeme súbor `grub.conf`.

Okopírujeme predchádzajúci odstavec od textu **title** a pridáme ho na koniec súboru. V novom odstavci upravíme tieto riadky:

Riadok začínajúci na **title** napríklad na `title Fedora Core - moje jadro`.

Potom upravíme riadok, začínajúci na slovo **kernel**. V ňom zadáme názov nového súboru `vmlinuz` tak, ako sme si ho pomenovali. Ostatné voľby ponecháme, lebo ich istotne dobre vytvoril inštalačný program pri inštalácii systému.

Vzor upraveného súboru `grub.conf` je na výpise č.4:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#           all kernel and initrd paths are relative to /, eg.
#           root (hd0,0)
#           kernel /boot/vmlinuz-version ro root=/dev/hda1
#           initrd /boot/initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Fedora Core (2.4.22-1.2115.nptl)
    root (hd0,0)
    kernel /boot/vmlinuz-2.4.22-1.2115.nptl ro root=LABEL=/ hdc=ide-scsi rhgb
    initrd /boot/initrd-2.4.22-1.2115.nptl.img

title Fedora Core moje jadro
    root (hd0,0)
    kernel /boot/vmlinuz-2.4.22-mojejadro ro root=LABEL=/ hdc=ide-scsi rhgb
    initrd /boot/initrd-2.4.22-1.2115.nptl.img
```

Súbor uložíme.

Tým sme v systéme zachovali aj pôvodné funkčné jadro a pridali jadro nové. Je to veľmi výhodné, lebo keď zistíme, že nové jadro nefunguje správne, stále máme možnosť použiť jadro staré a vytvorenie nového začať znova od začiatku.

Po reštarte systému sa v ponuke bootloadera objaví nová položka s názvom **Fedora Core - moje jadro**.

Stačí, aby sme na ňu presunuli kurzor a odklepnutím *Enteru* spustíme boot systému s novým jadrom.

V prípade, že sa nové jadro osvedčí a my chceme, aby sa spúšťalo samostatne bez toho, aby sme ho museli odklikávať, nastavíme v súbore `grub.conf` položku **default** na hodnotu 1. Tým zabezpečíme, že sa po určitom čase, stanovenom v položke **timeout**, automaticky spustí druhá položka menu (prečo druhá, keď zadáme číslo 1? Lebo prvá položka má číslo nula, druhá číslo 1, tretia číslo 2 atď.)

Úprava bootloadera LILO

Ak v našej distribúcii používame lilo, upravíme súbor `/etc/lilo.conf`.

Podobne ako v predchádzajúcom prípade, okopírujeme prvý odstavec, začínajúci na slovo **image** a pridáme ho na koniec súboru. Upravíme riadky **image**, kde napíšeme súbor nového jadra a **label**, kde definujeme pomenovanie položky, pod ktorou sa nové jadro bude prezentovať. Vzor upraveného súboru `lilo.conf` je na výpise č.6:

```
default=Linux
image=/boot/vmlinuz-2.4.22-1.2115.nptl
    label=Linux
    initrd=/boot/initrd.gz
    read-only

image=/boot/vmlinuz-2.4.22-mojejadro
    label=Linux-moje_jadro
    initrd=/boot/initrd.gz
```

read-only

Súbor uložíme.

Na rozdiel od bootloadera GRUB musíme uvedené zmeny v súbore *lilo.conf* aktivovať spustením príkazu

```
[root@doma linux-2.4] # lilo
```

Môžeme reštartovať.

V novej verzii LILO, ktoré má už podobu menu, podobne ako GRUB, sa objaví nová položka s príslušným názvom. Presunutím kurzora a odkliknutí na naštartuje nové jadro.

Ak používame staršiu verziu LILO, ktoré nemá podobu menu, môžeme pri zobrazení textu **boot:** na obrazovke tabulátorom zvoliť, ktorú položku chceme aktivovať.

Ak nám nové jadro vyhovuje, môžeme nastaviť jeho defaultné spúšťanie úpravou položky **default** na hodnotu *default=Linux-moje_jadro* v súbore */etc/lilo.conf*.

Po tejto úprave nesmieme zabudnúť znova spustiť príkaz **lilo**, ktorý dané zmeny zaktivuje.

Súbor .config

V prípade, máme viac počítačov, na ktorých chceme vykonať identickú úpravu jadra, nemusíme prechádzať príslušné voľby konfiguračného nástroja.

Vtedy stačí, ak na všetky upravované počítače prekopírujeme súbor *.config*, ktorý vznikol po konfigurácii jadra na prvom počítači. Potom spustíme samotný preklad zdrojových kódov a modulov.

Tak verím, že sa nám prvý preklad jadra zdárne vydaril a my sme hrdo postúpili o jednu priečku v znalostiach Linuxu.

Miroslav Oravec

Linux prakticky ako server aj ako desktop / 15.časť

Tak ako minule, tak aj teraz je táto časť seriálu spoločná pre obidve skupiny – pre serveristov aj pre desktopistov.

A prečo?

No to pochopíme z dnešnej lekcie.

Prednedávnom sme potrebovali pre jednu novú kancelársku silu pripraviť počítač. Nič náročného, len aby zvládol bežnú kancelársku agendu, ako je písanie textov a tvorba tabuliek, no a ešte aby zvládol pripojenie k Internetu. Lenže nový počítač nebol. A tak sme v šuplíkoch ponachádzali staršie diely, ktoré pozostávali z predchádzajúcich repasácií. Dali sme to dohromady v tejto zostave: Pentium I/ 75 MHz, 24 MB RAM, 540 MB pevný disk, disketová mechanika, sieťová karta a grafika S3 4 MB PCI. Skutočne nič moc... (Radšej) nemenovaný operačný systém by sa vliekol, nešlo by urobiť nič poriadneho, a na dôvažok vedúci nemal žiadne finančné prostriedky na položke „Software“. Keďže sme už mali v sieti slušný linuxový server, nastal čas na vytvorenie linuxovej desktopovej stanice v podobe tenkého klienta...

Ako sme to urobili, to si dnes ukážeme. Uvidíte, že to bude celkom jednoduché.

Základné pojmy

Aj keď by sme už princíp tenkého klienta mali poznať, pripomeňme si v stručnosti jeho podstatu:

Na strane linuxového hlavného počítača (ktorému budeme hovoriť **aplikačný server**) prebieha všetok operačný výkon – programy (aplikácie) ležia na jeho pevnom disku a spúšťajú sa v jeho operačnej pamäti. Výsledky aplikácie sa posielajú na obrazovku linuxového tenkého klienta, a ten zase posiela odozvy klávesnice a myši k serveru. Z tohto dôvodu nie sú kladené veľké nároky na výkonnosť desktopového klienta a môže to byť aj starší počítač. Je samozrejmé, že k jednému serveru môže byť pripojených viac tenkých klientov. Pre jednoduchosť budeme tenkému linuxovému desktopovému klientovi hovoriť **X-terminál**. Používateľa, sediaceho za X-terminálom, budeme nazývať **vzdialený používateľ**, používateľa, sediaceho za klávesnicou (konzolou) aplikačného servera budeme nazývať **lokálny používateľ**.

Sieť

Okrem aplikačného servera a X-terminálu musíme mať k dispozícii sieť. Nie je vyslovene nutné, že táto sieť musí byť čo najrýchlejšia a najmodernejšia. Vystačíme si aj zo staršou sieťou o rýchlosti 10 Mbps, a je jedno, či bude založená na koaxiálnom kábli alebo na štrukturovanej kabeláži. Prax potvrdzuje, že 10Mbps sieť postačuje pre približne 40 tenkých klientov.

Samozrejme, že môžeme úspešne použiť aj bezdrôtovú wi-fi sieť. Tá podľa normy 801.11b pracuje do rýchlosti 11 Mbps a podľa normy 801.11g do rýchlosti 55 Mbps. Technické prostriedky obidvoch noriem sú už veľmi cenovo prístupné a ako sa sprevádzkujú pod Linuxom, to sme si už v minulých častiach – desktopovej aj serverovej – ukázali.

Softvérové vybavenie

Na strane aplikačného servera je nainštalovaná *Fedora Core1* (FC1), na stranu X-terminálu sme dali takisto FC1. Zatiaľ čo aplikačný server je nainštalovaný v pozícii plnokrvného servera vrátane systému X-Window a X aplikácii, X-terminál je nainštalovaný v čo najminimálnejšej desktopovej konfigurácii. Je však iba na nás, akú distribúciu Linuxu použijeme, princípy nastavenia sú rovnaké, môžu sa líšiť v malých detailoch v cestách k jednotlivým konfiguračným súborom.

Nastavenie Linuxu

Prvé, čo musíme mať na dvojici aplikačný server a X-terminál funkčné, je vzájomná komunikácia po počítačovej sieti. Ak pripájame X-terminál do už fungujúcej siete, vyberieme vhodnú IP adresu a masku, prípadne požiadame o pomoc správcu siete. Ak budujeme novú X-terminálovú sieť od základov, počítame s vhodnými IP adresami. Funkčnosť siete overíme príkazom *ping*. Ak sa nám pingy vracajú, je sieť v poriadku.

Správca prihlásenia – display manager

Pamätáte si na úplne prvé lekcie o prostredí X Window? V teórii o činnosti tohto grafického prostredia sme si povedali, ako to približne funguje. K všetkým tým rôznym správcam okien – *window managerom* ešte prináleží

aj jeden iný správca, ktorého úlohou je zabezpečiť prihlásenie sa do grafického systému. Tento správca prihlásenia sa nazýva *display manager*. Má grafickú podobu (akože ináč) a niekoľko konfiguračných súborov.

Ako išiel vývoj prostredia X Window, vyvíjali sa aj rôzni správcovia prihlásenia.

V podstate existujú tieto najznámejšie display manager-e:

- Ø **x**dm – **X** Display Manager
- Ø **g**dm – **G**nome Display Manager
- Ø **k**dm – **K** Display Manager

xdm je najstarší z nich. Je síce súčasťou každej väčšej linuxovej distribúcie, ale dnes sa už používa iba na menších desktopových distribúciách Linuxu, kde sa používajú iní – menší správcovia okien.

gdm – ako už z názvu vyplýva, je súčasťou prostredia *Gnome* a **k**dm zase súčasťou prostredia *KDE*.

Poznámka:

Je naozaj na nás, ktorý display manager si vyberieme. Všetky dokážu spustiť požadované grafické prostredie, či už Gnome alebo KDE. Preto nie je vyslovene nutné, že keď chceme používať na X-termináli KDE, tak musíme používať kdm!

Výber display managera

V našej situácii sme sa rozhodli vybrať *gdm* – *Gnome Display Manager*. Nielen že je vzhľadovo príjemný, ale je pomerne jednoduchý na konfiguráciu. Preto tu uvedený postup je šitý na tento manažér. V závere je spomenutý stručný postup na nastavenie ostatných dvoch display manažérov.

Nastavenie aplikačného servera

Základom správnej činnosti tenkého klienta je nastavenie aplikačného servera. Nastavenie môžeme zhrnúť do niekoľkých krokov:

- Ø inštalácia a nastavenie základu systému Linux
- Ø inštalácia a nastavenie systému X Window tak, aby bol schopný bežať na konzole aplikačného servera
- Ø inštalácia a nastavenie X aplikácií, ktoré budeme používať na X-termináli
- Ø pridanie jednotlivých (aj vzdialených) používateľov
- Ø nastavenie a odskúšanie siete

a prípadne (voliteľne)

- Ø nastavenie a odskúšanie tlačiarní
- Ø nastavenie pripojenia do Internetu
- Ø nastavenie ďalších serverových úloh, napr. web server, proxy server, poštový server a podobne.

a nakoniec

- Ø úprava konfiguračných súborov display manažéra
- Ø úprava konfiguračných súborov aplikačného servera

Možno nás prekvapí, že čo všetko po jednom Linuxovom serveri požadujeme, ale verte, že to nie je nič nemožné a Linux to všetko zvládne.

Poznámka:

To, či to všetko ponecháme na jednom jedinom stroji, záleží iba od počtu pripojených X-terminálov. Ak máme malý počet vzdialených tenkých klientov, napr. 10 – 15 v menšej firmičke, tak postačí iba jeden linuxový server na všetky serverové úlohy. Ak však prevádzkujeme 400 X-terminálov (ako vzor použijem firmu Largo, USA), tak je efektívnejšie a bezpečnejšie použiť na prevádzku X-terminálov jeden (až niekoľko) samostatných aplikačných serverov a pre ostatné serverové úlohy použijeme iný samostatný linuxový server.

Pozor!

Všetky úkony na aplikačnom serveri vykonávame ako root!

Ak máme nastavený základ operačného systému Linux, prejdeme k nastaveniu X Window systému. To, či systém naozaj beží, overíme prihlásením sa na konzole aplikačného servera príkazom *startx*. Aplikácie, s ktorými chceme pracovať na X-termináloch, musíme mať nainštalované a nastavené na aplikačnom serveri. Je logické, že konkrétnu aplikáciu stačí mať nainštalovanú pre všetkých vzdialených aj lokálnych používateľov **iba raz**! Zvyšok zabezpečí jadro Linuxu, veď je to viacúčelový a viacpoužívateľský systém! Nesmieme zabudnúť pridať do systému všetkých používateľov. Pridelíme loginy a nastavíme heslá. Zároveň každému používateľovi upravíme základný vzhľad grafického prostredia podľa jeho potrieb, vrátane pozadia, rozlíšenia a iné. Ostatné si môže nastaviť používateľ sám priamo cez X-terminál. Odkúšame *pingom* sieť a skontrolujeme, či všetko funguje ako má. Nesmieme zabudnúť na internetové nastavenia, ako je brána, DNS server a podobne, aby sme mohli z X-terminálov používať aj internetové pripojenie. Toto všetko odkúšame na konzole aplikačného servera. Ak to bude fungovať na konzole, tak s najväčšou pravdepodobnosťou to bude fungovať aj na X-termináli. Ak však toto neoveríme, nebudeme mať istotu, kde je vlastne chyba, či na serveri, či v sieti alebo v X-termináli.

Úprava konfiguračných súborov *display managera*

Konfiguračné súbory *gdm* display manažéra sa nachádzajú v adresári */etc/X11/gdm/*. Tu sa nachádza veľmi dôležitý súbor *gdm.conf*.

Otvoríme tento súbor vo vhodnom textovom editore. Vyhľadáme sekciu **XDMCP** – výpis č.1:

```
# GDM Configuration file.  You can use gdmsetup program to graphically
# edit this, or you can optionally just edit this file by hand.  Note that
# gdmsetup does not tweak every option here, just the ones most users
# would care about.  Rest is for special setups and distro specific
.
.(skrátene)
.
.

# Have fun! - George

[daemon]
# Automatic login, if true the first local screen will automatically logged
.
.(skrátene)
.
[security]
# If any distributions ship with this one off, they should be shot
.
.(skrátene)
.
[xdmcp]
# Distributions: Ship with this off.  It is never a safe thing to leave
# out on the net.  Setting up /etc/hosts.allow and /etc/hosts.deny to only
# allow local access is another alternative but not the safest.
# Firewalling port 177 is the safest if you wish to have xdmcp on.
# Read the manual for more notes on the security of XDMCP.
Enable=true
# Honour indirect queries, we run a chooser for these, and then redirect
# the user to the chosen host.  Otherwise we just log the user in locally.
HonorIndirect=true
.
.(skrátene)
.
[gui]
# The 'theme'.  By default we're using the default gtk theme
.
.(skrátene)
.
[greeter]
# Greeter has a nice title bar that the user can move
TitleBar=false
.
.(skrátene)
.
```

Štandardne je vzdialené prihlasovanie zo vzdialených terminálov zakázané, preto upravíme položku

Enable=false

na

Enable=true

Súbor uložíme. Éto vsjo! Ved' som vám sľúbil, že to bude veľmi jednoduché!

Úprava konfiguračných súborov aplikačného servera

Už sme si povedali, že na zabezpečenie prihlásenia sa do grafického prostredia slúži správca prihlásenia display manager. Preto ako prvé musíme zabezpečiť, aby bol display manager spustený. Ak sme si vybrali *Gnome Display Manager*, budeme ho spúšťať pomocou príkazu **gdm**, teda napríklad

```
[root@rubin root]# /usr/bin/gdm
```

alebo jednoducho

```
[root@rubin root]# gdm
```

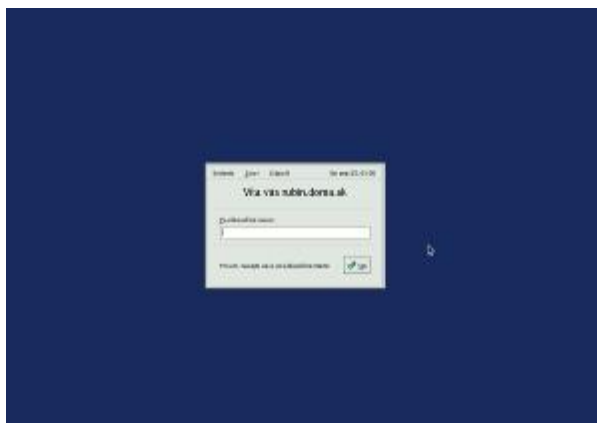
Najlepšie by bolo, keby sa *gdm* spúšťalo automaticky pri štarte operačného systému. To môžeme zabezpečiť niekoľkými spôsobmi.

Ø do súboru */etc/rc.d/rc.local* pripíšeme riadok:

```
/usr/bin/gdm
```

Ø v príslušnom runleveli vytvoríme symbolickú linku a jednoduchý skriptík na spúšťanie služby (vzor, ako sa to robí, sme si už ukázali a nebudeme sa k tomu vracáť).

Po týchto úpravách spustíme (ručne alebo automaticky) Gnome Display Manager. Na obrazovke aplikačného servera sa objaví prihlasovacie okno – obr.č.2:



Nič nevyplňujeme, nič nezadáваме, neprihlasujeme sa! Len sme overili funkčnosť display manažéra *gdm*.

Nastavenie X-terminálu

Aj nastavenie X-terminálu môžeme rozdeliť do niekoľkých krokov:

Ø inštalácia a nastavenie základu systému Linux

- Ø inštalácia a nastavenie systému X Window tak, aby bol schopný bežať na konzole X-terminálu
- Ø nastavenie a odskúšanie siete
- Ø úprava konfiguračných súborov X-terminálu
- Ø prevádzka X-terminálu

Na rozdiel od aplikačného serveru inštalácia X-terminálu pozostáva z minima súčastí. Nesmieme zabudnúť, že aj v desktopovej podobe je dobré používať odkladací swapovací súbor. Ten by mal mať veľkosť rovnajúcu sa dvojnásobku operačnej pamäte X-terminálu, takže my sme zvolili 50 MB pre swapovací priestor. Pri inštalácii je dôležité, aby sme nainštalovali systém X Window, teda iba X-server vrátane príslušných komponentov. Neinštalujeme žiadne grafické nadstavby, ako je Gnome, KDE a podobne! Neinštalujeme ani žiadne aplikácie! Nebudeme to potrebovať (tie budeme spúšťať z aplikačného serveru)! (Najzákladnejšia inštalácia X-serveru sa vojde na pevný disk do 80 MB, ba dokonca aj na jednu - jedinú disketu!).

Poznámka:

Začiatočník môže mať zo začiatku problém rozhodnúť, ktoré komponenty treba alebo netreba inštalovať. Nič sa nestane, keď na skúšku zadáme inštaláciu pre desktop alebo plnú inštaláciu – pokiaľ ovšem máme dostatočne veľký pevný disk! Ak nám X-terminál funguje, môžeme pristúpiť k odstraňovaniu jednotlivých programov a utilít, až sa dopracujeme k rozumnej veľkosti – aj ja som to tak robil).

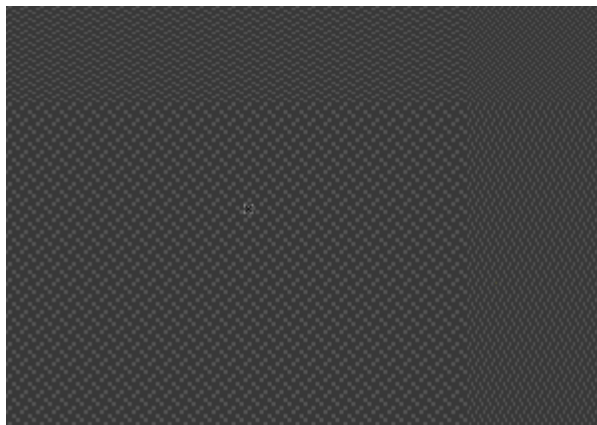
Zatiaľ počítač k sieti nepripojujeme!

Na počítači, ktorý bude slúžiť ako X-terminál máme iba jedného používateľa a tým je *root*. Iných používateľov nepridávame (tí sú už nadefinovaní na aplikačnom serveri). Heslo pre roota si môžeme zvoliť ľubovoľne, ale z bezpečnostného hľadiska by to nemalo byť to isté heslo, ako má root na aplikačnom serveri!).

Pozor!

Aj nastavenia X-terminálu vykonávame ako root!

Po inštalácii vyskúšame spustiť X-server príkazom *startx*. Ak sme nainštalovali plný desktop, tak sa objaví klasické grafické prostredie. Ak sme inštalovali iba minimum – teda len X-server, zobrazí sa šedá obrazovka s kurzorom myši – obr.č.3:



X-server ukončíme stlačením klávesov *Ctrl-Alt-Backspace*. (Niekdý je nutné tento trojhmat opakovať viackrát...).

Vrátime sa do režimu príkazového riadku. (V prípade, že X-terminal neštartuje do príkazového riadku, upravíme súbor */etc/inittab*)

Pripojíme konektor od sieťového pripojenia a nastavíme sieť. Po nastavení overíme funkčnosť siete príkazom *ping*. Ak je sieť priechodzia, prikrčíme k pripojeniu X-terminálu k aplikačnému serveru.

Pripojenie k aplikačnému serveru

Pripojenie tenkého klienta k aplikačnému serveru vykonáme pomocou príkazu

X -query ip_adresa_servera

Teda ak je IP adresa nášho aplikačného servera 192.168.10.1, príkaz bude

```
[root@rubin root]# X -query 192.168.10.1
```

Ak sme všetko nastavili správne, na obrazovke nášho X-terminálu sa objaví prihlasovacie okno, podobné tomu na obrázku č.2!

Heuréka!

Spojenie funguje! Aj tu X-server killneme trojhmatom *Crtl-Alt-Backspace*.

Úprava konfiguračných súborov X-terminálu

Doteraz sme sa na X-termináli logovali ako root. My vieme, že to nie je správne, lebo to môže viesť k nechcenému poškodeniu systémových súborov. Zároveň si musíme uvedomiť, že root na X-termináli nie je totožný s rootom na aplikačnom serveri. Sú to v podstate dve rozdielne existencie, aj keď to možno je fyzicky tá istá osoba!

A vôbec, my sa nepotrebujeme do systému Linuxu na X-terminále logovať. To bolo iba vtedy, keď sme robili prvotné nastavenia. Pre skutočnú prácu sa potrebujeme nalogovať do aplikačného servera – práve pomocou X-terminálu.

Pozor!

My si musíme uvedomiť, že logovacia obrazovka X-terminálu je v skutočnosti vzdialená obrazovka aplikačného servera. V jednoduchosti – monitor, klávesnica a myš X-terminálu sú akoby vzdialené súčasti aplikačného servera! Pamätajte na to!

Preto musíme zabezpečiť, aby sa logovacie okno spustilo hneď po štarte systému Linux (použijeme podobný spôsob ako pri štarte display managera na aplikačnom serveri).

Vytvoríme si štartovací skript, ktorý pomenujeme *xterminal*. Jeho obsah bude vyzerat' takto:

```
#!/bin/sh
Echo "Spustam X terminal"
/usr/X11R6/bin/X -query 192.168.10.1
```

Súbor uložíme na vhodné miesto, napr. */etc/X11/* a upravíme jeho atribúty príkazom

```
[root@rubin X11]# chmod 755 xterminal
```

Potom upravíme súbor */etc/rc.d/rc.local*, kde na jeho koniec pripíšeme riadok:

```
/etc/X11/xterminal
```

Súbor uložíme.

Vykonáme reštart celého systému X-terminálu.

Práca na X-terminále

Ak sme všetko riadne nastavili, po naboťovaní celého systému sa automaticky spustí X-terminál. Objaví sa úvodná logovacia obrazovka tak, ako ju poznáme z obrázku č.2. Na celej obrazovke je pre nás najdôležitejšie okno v prostriedku obrazovky. Zadáme meno – obr.č.4:



odentrujeme a zadáme heslo. Zadáme také meno a heslo, ktoré je nadefinované v aplikačnom serveri!
Po odsúhlasení sa vykoná klasický štart grafického prostredia – obr.č.5:

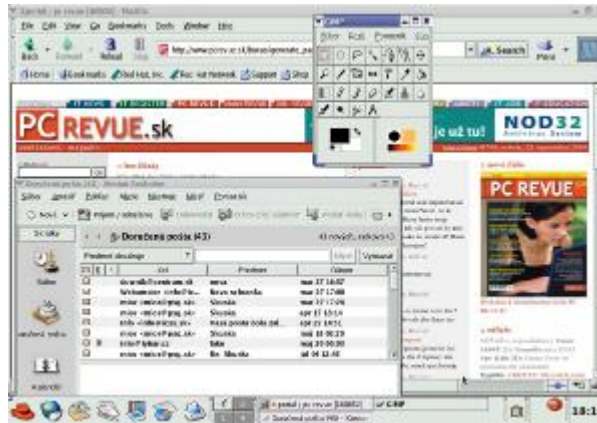


Túto obrazovku už poznáme, je to klasický štart grafického prostredia Gnome alebo KDE na distribúcii Fedora Core 1 (tí, ktorí použijú inú distribúciu, dostanú inú obrazovku).

Po kompletnom spustení grafického prostredia, ktoré bolo danému používateľovi vopred nastavené na aplikačnom serveri, sa na obrazovke monitora X-terminálu zobrazí pracovná plocha daného prostredia. Len pripomínam, že to môže byť kľudne aj KDE, aj keď sme na logovanie použili Gnome Display Manager. A tak je to aj v našom prípade – obr.č.6:



Teraz už môžeme normálne pracovať. To, že aj na slabšom stroji môžeme používať KDE, mať otvorené viaceré okná, ako je internetový prehliadač Mozilla so stránkou PC revue, program na čítanie elektronickej pošty Ximian Evolution a ešte k tomu grafický program Gimp, dokazuje obrázok č.7 (a to som chcel otvoriť ešte viac okien, ale už sa mi nevopchali na obrazovku):



Musíme uznať, že keby sme to chceli urobiť na sólo počítači ako desktop, potrebovali by sme podstatne silnejší stroj!

Nastavenie správcov prihlásenia xdm a kdm

Teraz si v stručnosti ukážeme, ako sa nastavujú ostatné dva display manažery. Princípy sú podobné, líšia sa len v jednotlivých konfiguračných súboroch.

xdm

Aby sme mohli používať pre vzdialených používateľov správcu prihlásenia *xdm*, musíme v adresári */etc/X11/xdm/* upraviť tieto súbory (na aplikačnom serveri):

- Ø *Xaccess*
- Ø *xdm-config*

Súbor *Xaccess*

Tento súbor vyeditujeme vhodným editorom a vyhladáme približne 40.riadok – výpis č. 8 (žiadaný riadok je zvýraznený):

```
# $XConsortium: Xaccess,v 1.5 91/08/26 11:52:51 rws Exp $
.
.(skrátene)
.
# *                                #any host can get a login window
.
.(skrátene)
.
#*                                CHOOSER %hostlist #
```

Našou úlohou je odkomentovať, teda odstrániť znak # na začiatku daného riadku, aby riadok vyzeral takto:

```
*                                #any host can get a login window
```

Tým sme povolili, aby ľubovoľný vzdialený počítač mohol dostať logovaciu obrazovku. Súbor uložíme.

Súbor *xdm-config*

Aj tento súbor vyeditujeme. Prejdeme na posledné tri riadky na konci súboru (sú zvýraznené na výpise č.9):

```
! $XConsortium: xdm-conf.cpp /main/3 1996/01/15 15:17:26 gildea $
! $XFree86: xc/programs/xdm/config/xdm-conf.cpp,v 1.6 2000/01/31
19:33:43 dawes Exp $
DisplayManager.errorLogFile: /var/log/xdm-errors
DisplayManager.pidFile:      /var/run/xdm-pid
DisplayManager.keyFile:      /etc/X11/xdm/xdm-keys
```

```
DisplayManager.servers:      /etc/X11/xdm/Xservers
DisplayManager.accessFile:   /etc/X11/xdm/Xaccess
.
.(skrátené)
.

! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:  0
```

Tentokrát na začiatok posledného riadku vložíme výkričník – takto:

! DisplayManager.requestPort: 0

Súbor uložíme.

Správca prihlásenia *xdm* spúšťa príkazom **xdm**.

Tak ako pri *gdm*, tak aj tu môžeme príslušným spôsobom upraviť štartovacie skripty aplikačného servera, aby sa *xdm* spúšťalo automaticky pri štarte systému.

Na strane X-terminálu vykonáme tie isté zmeny ako pri *gdm*.

kdm

Tento správca prihlásenia používa tieto konfiguračné súbory:

- Ø Xaccess
- Ø kdmrc

Obidva sa nachádzajú v tom istom adresári */etc/X11/xdm/*.

Súbor *Xaccess* upravíme tak, ako pri *xdm*.

Poznámka:

Možno sa niekde dočítate, že na konfiguráciu kdm sa používajú iné súbory v iných adresároch. Nedajte sa zmiasť. To sú len linky na tu spomínané súbory!

Súbor kdmrc

Tento súbor otvoríme v editore a prejdeme na sekciu [Xdmcp] – výpis č.10:

```
# KDM configuration example.
# Note, that all comments will be lost if you change this file with
# the kcontrol frontend.
.
.(skrátené)
.
[Desktop0]
BackgroundMode=VerticalGradient
.
.(skrátené)
.
[General]
# If "false", KDM won't daemonize after startup. Use this, if you start
.
.(skrátené)
.
[Xdmcp]
# Whether KDM should listen to XDMCP requests. Default is true.
Enable=true
# The UDP port KDM should listen on for XDMCP requests. Don't change
the 177.
#Port=177
.
.(skrátené)
```

```
.  
# Greeter config for 1st local display  
[X-:0-Greeter]  
# See above  
#PreselectUser=Default  
# See above  
#DefaultUser=johndoe
```

Vyhľadáme položku **Enable** a nastavíme ju na **Enable=true** (štandardne je nastavená na *false*). Súbor uložíme.

Správcu prihlásenia *kdm* spúšťame príkazom **kdm**.

Aj tu môžeme upraviť štartovacie skripty aplikačného servera, aby sa *kdm* spúšťalo automaticky pri štarte systému.

Na strane X-terminálu vykonáme tie isté zmeny ako pri *gdm*.

Drobnosti

Možno vás napadne otázka, akože sme to inštalovali Linux na X-terminál, keď tento počítač nemá mechaniku CD-ROM?

Aj toto je veľmi jednoduché – vzali sme mechaniku z iného počítača, na chvíľu pripojili k X-terminálu a po nainštalovaní sme ju zase vrátili späť.

Že to nie je pravé linuxové?

Dobre, tak si môžeme vyskúšať inštaláciu Linuxu po sieti, ale o tom nabadúce.

Linux prakticky ako server aj ako desktop/ 16.časť

V minulých častiach sme sa venovali prekladu jadra a vytvoreniu tenkého klienta. Boli to zaujímavé témy, ale ešte častejšie sa používateľovi stáva, že potrebuje vykonávať určitú činnosť v určitom čase. No a keďže posadiť sa s hodinkami v ruke za klávesnicu počítača a v daný okamžik, napr. o pol jednej v noci na Vianoce spustiť žiadaný príkaz je naozaj, ale naozaj veľmi nelinuxovské, existujú v systéme Linuxu dva nástroje na časové spúšťanie úloh. A dnes sa týmto nástrojom budeme venovať.

Rozdelenie nástrojov

Aby sme správne pochopili, ako sa narába so spomínanými nástrojmi, rozdelíme si ich podľa spôsobu činnosti. Aj z bežného života vieme, že určitú činnosť môžeme vykonať iba raz v živote alebo môžeme chcieť, aby sa táto činnosť vykonávala opakovane a pravidelne.

Tak isto je to aj v Linuxe. Na opakované spúšťanie danej úlohy slúži program **cron**, na jednorázové spustenie konkrétnej úlohy slúži program **at**.

Cron

Cron je veľmi mocný nástroj, ktorý sa nachádza v každej linuxovej distribúcii a je aktívny pri každom štarte systému. Môžeme teda povedať, že je to služba. Začiatočníci si ani poriadne neuvedomujú prítomnosť cron-u, ale on v systéme je a poctivo vykonáva stanovené úlohy. Nezáleží pritom, či sa jedná o server alebo desktop. Už z inštalácie je *cron* nastavený tak, že vykonáva údržbu operačného systému, maže nepotrebné dočasné súbory a podobne. Z používateľského hľadiska môžeme *cron* využívať na automatické vykonávanie záloh dát, odosielanie a prijímanie pošty, odosielanie sms-iek o stave servera alebo jednoducho odoslať upozornenie, že máme ísť na poradu.

Podstata činnosti programu cron

Pravidelne, každú minútu si program cron pozrie súbory s úlohami, a ak sa v niektorom z nich nachádza zaznamenaný časový údaj, zodpovedajúci skutočnému času, definovanú úlohu, priradenú časovému údaju, vykoná.

Súbory programu cron

Povedali sme si, že program *cron* je v podstate službou. Túto službu zabezpečuje démon *crond*. Tak ako každú inú službu, tak aj túto môžeme ovládať príkazom **service crond** s príslušným parametrom. Čo a kedy má cron vykonať, to sa nachádza v súboroch s úlohami, ktorým môžeme hovoriť aj definičné súbory. Definičné súbory môžu byť takéhoto typu:

- Ø systémový súbor
- Ø používateľský súbor
- Ø systémové adresáre

Systémový súbor sa volá **crontab** a je uložený v adresári */etc/*.

Používateľské súbory majú názov podľa mena používateľa, ktorý ich vytvoril. Meno je také, ako je zadefinované v súbore */etc/passwd*. Ich uloženie nie je v každej distribúcii rovnaké, napr. v distre Red Hat/Fedora je to súbor */var/spool/cron/<meno_používateľa>*, v SUSE je to */etc/cron/tabs/<meno_používateľa>* a v Slackware to je */var/spool/cron/crontabs/<meno_používateľa>*.

Predstavme si používateľa s menom **mior**, pracujúcom na distribúcii Fedora Core. Jeho definičný súbor pre program cron bude */var/spool/cron/mior*. Často (a nesprávne) sa týmto súborom hovorí používateľské crontab súbory. Aby sme si ich my nemýlili so skutočným súborom */etc/crontab*, budeme používateľské súbory nazývať „používateľské cron-tabuľky“.

Dôležité je, že systémový */etc/crontab* aj používateľská cron-tabuľka, napríklad */var/spool/cron/mior*, majú rovnakú štruktúru.

Štruktúra cron-tabuliek

Pozrime sa na súbor `/etc/crontab`. Na výpise č.1 je jeho obsah:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Súbor je rozdelený do dvoch častí:

Prvá časť skriptu popisuje všeobecné nastavenia. Premenná `SHELL` určuje použitie príkazový interpreter, v našom prípade je to `bash`.

`PATH` je cesta pre program `cron`, aby vedel, kde sa nachádzajú jeho pomocné programy.

`MAILTO` sa odkazuje na používateľa, ktorému má zasielať e-mail. Obsahom týchto e-mailov je výstup programu, ktorý bude `cronom` spustený. Jednoduchšie povedané – ak program alebo príkaz, spustený `cronom` v danom čase používa štandardný výstup (na obrazovku), obsah tohto výstupu bude poslaný do emailovej schránky používateľa, v tomto prípade `roota`.

`HOME` označuje domovský adresár, v tomto prípade koreň celého adresárového stromu.

Druhá časť súboru obsahuje definície času a požadovanej úlohy.

Skladá sa zo siedmich položiek a jednotlivé položky sú oddelené medzerou:

Minúta	Hodina	Deň v mesiaci	Mesiac	Deň v týždni	Používateľ	Príkaz (úloha)
--------	--------	---------------	--------	--------------	------------	----------------

Položka **Minúta** (angl. *minute*) môže nadobúdať hodnoty od 0 do 59 a popisuje, v ktorej minúte bude príkaz (úloha) spustená.

Položka **Hodina** (angl. *hour*) môže nadobúdať hodnoty od 0 do 23 a popisuje, v ktorej hodine bude príkaz (úloha) spustená. Uvedomme si, že sa jedná o 24 hodinový formát a 0 (nula) symbolizuje polnoc!

Položka **Deň v mesiaci** (angl. *day of month - dom*) môže nadobúdať hodnoty od 1 do 31 (podľa mesiaca) a popisuje, v ktorom dni mesiaca bude príkaz (úloha) spustená.

Položka **Mesiac** (angl. *month*) môže nadobúdať hodnoty od 1 do 12 a popisuje, v ktorom mesiaci bude príkaz (úloha) spustená.

Položka **Deň v týždni** (angl. *day of week - dow*) môže nadobúdať hodnoty od 0 do 7 a popisuje, v ktorom dni v týždni hodine bude príkaz (úloha) spustená. Hodnote 0 (nula) a hodnote 7 zodpovedá nedeľa, hodnote 1 zodpovedá pondelok a tak ďalej až hodnote 6 zodpovedá sobota.

Položka **Používateľ** (angl. *user*) obsahuje meno používateľa, pod ktorého právami sa bude príkaz (úloha) vykonávať. Táto položka nie je povinná a môže byť (a aj v používateľských cron-tabuľkách naozaj je) vynechaná.

Položka **Príkaz** (angl. *command*) obsahuje názov príkazu, programu alebo skriptu, ktorý sa bude v daný časový okamžik spúšťať.

Malý príklad

Predstavme si, že chceme práve v piatok 24.decembra o 17-tej hodine a 25-tej minúte spustiť skript s názvom **zaloha**, nachádzajúci sa v adresári `/usr/bin/`, ktorý s právami `roota` vykoná archiváciu všetkých konfiguračných súborov v našom systéme (obsah tohto skriptu teraz nie je podstatný).

Potom by zápis v súbore `/etc/crontab` vyzeral takto:

```
25 17 24 12 5 root /usr/bin/zaloha
```

Upravený súbor uložíme a môžeme si spokojne sadnúť k štedrovečernému stolu, lebo zálohy sa vykonajú aj bez našej prítomnosti.

Musíme si ale uvedomiť niekoľko skutočností:

- *cron* je periodický program, takže neberie v úvahu rok. Všimnime si, že v definíciách času nie je rok uvedený. To znamená, že sa všetko vykoná s ročnou periódou.
- Všetky časové podmienky musia byť súčasne splnené! To znamená, že sa program *zaloha* síce v tomto roku naozaj vykoná, lebo 24.decembra prípadne na piatok, ale keďže nasledujúci rok 24.decembra (2005) prípadne na sobotu, príkaz *zaloha* sa nevykoná, lebo nie je splnená podmienka dňa v týždni! (To bude splnené až v roku 2010, kedy deň 24. decembra zase prípadne na piatok - ak neveríte, pozrite sa do kalendára).

Prvých päť časových položiek môže okrem číselného údaju obsahovať niekoľko zvláštnych znakov - *modifikátorov*, ktoré si postupne vysvetlíme.

Modifikátor *

Najdôležitejším modifikátorom je znak * (hviezdica), ktorá symbolizuje akúkoľvek hodnotu. Pod slovom „akúkoľvek“ môžeme pre tento prípad rozumieť „každú“ hodnotu.

Ako by sme teda upravili vyššie spomínaný príklad tak, aby sa vykonával každoročne? Stačí, ak kritickú položku *deň v týždni* nahradíme hviezdikou:

```
25 17 24 12 * root /usr/bin/zaloha
```

To znamená, že nech 24. december ľubovoľného roku prípadne na akýkoľvek deň v týždni, príkaz */usr/bin/zaloha* sa v daný čas vykoná!

Možno si povieme, že vykonávať zálohy raz ročne je prinajmenšom trestuhodné a tak to budeme chcieť vykonávať raz mesačne a to každého 24-tého dňa v mesiaci.

No nič jednoduchšie! Položku *mesiac* takisto nahradíme hviezdikou:

```
25 17 24 * * root /usr/bin/zaloha
```

Ale robiť zálohy raz za mesiac tiež nie je to pravé orechové... Preto doporučujeme vykonávanie raz do týždňa, napríklad v stredu. Potom upravíme zápis takto:

```
25 17 * * 3 root /usr/bin/zaloha
```

Vymenili sme položku *deň v mesiaci* za hviezdicu, lebo streda v týždni môže pripadnúť na ľubovoľný dátum. A namiesto *dňa v týždni* sme vložili trojku, ktoré predstavuje stredu.

No ale čo ak máme citlivé dáta a tie chceme archivovať každý deň o 17,25 hodine? Potom bude zápis vyzeráť takto:

```
25 17 * * * root /usr/bin/zaloha
```

Položky *deň v mesiaci*, *mesiac* a *deň v týždni* nahradíme hviezdikou, zostane iba *minúta a hodina*.

Ostatné modifikátory

Môže sa stať (a aj sa veľmi často stáva), že v našom systéme potrebujeme určité úlohy plniť častejšie ako raz za deň.

Takým extrémom môže byť pravidelné sťahovanie pošty každú minútu. Nech sa skript pre sťahovanie pošty nazýva */usr/bin/posta*, potom zápis bude takýto:

```
* * * * * root /usr/bin/posta
```

Preložené do našej reči to hovorí: každú minútu každej hodiny každého dňa každého mesiaca v každý deň v týždni sa vykoná stanovený príkaz.

Ak sme ale pripojení do internetu pomocou modemu, potom je ekonomickejšie sťahovať poštu len každé dve hodiny. Zápis pre túto činnosť bude

```
* */2 * * * root /usr/bin/posta
```

Modifikátor „/“ (lomka) udáva krok a značí „každý (časový úsek)“, teda zápis **/2* v položke *hodina* znamená každé dve hodiny.

Možno nám bude stačiť, aby sa pošta sťahovala iba v dennej dobe, v noci to nie je potrebné. Vtedy upravíme zápis takto:

```
* 6-18/2 * * * root /usr/bin/posta
```

Zápis hovorí, že úloha sa spustí každé dve hodiny v dobe od 6. hodiny ránej do 18. hodiny večernej.

Modifikátor „-“, značí „od – do“.

Predsa by len bolo dobre, keby sme stiahli poštu aj v noci, napríklad o 22. hodine pre nočného vrátnika. Zápis upravíme takto:

```
* 6-18/2,22 * * * root /usr/bin/posta
```

Položka *hodina* v tvare **6-18/2,22** znamená každé dve hodiny v dobe od 6. hodiny ránej do 18. hodiny večernej a ešte aj o 22. hodine. Modifikátor „“,“ zlučuje viac hodnôt.

Pozor!

Pri tomto zápise nesmieme použiť medzeru! Tá slúži ako oddeľovač jednotlivých položiek!

Vytváranie cron tabuliek

My sme si už hovorili, že každý používateľ si môže pre cron vytvoriť svoju vlastnú tabuľku. Táto tabuľka je uložená na špecifickom mieste, napríklad */var/spool/cron/mior*. Aby ju nebolo nutné vytvoriť a potom uložiť na príslušné miesto ručne, môžeme použiť program *crontab*.

Pozor!

Nezmýľme si program *crontab* so súborom *crontab*, to nie je to isté!

V Linuxe je bežné (hovorím bežné, nie normálne!), že sa programy aj súbory nazývajú rovnako.

Program *crontab* slúži na správu cron tabuliek. Má tieto parametre:

- Ø **-e** - editovanie súboru
- Ø **-l** - výpis súboru
- Ø **-r** - zmazanie súboru

Program *crontab* volá externý editor textu, ktorý je v systéme zadefinovaný v premennej *EDITOR*.

Spravidla to býva program *vi* alebo jeho obdoba *vim*.

Predstavme si, že sme používateľ *mior* a chceme vytvoriť svoju cron tabuľku, aby sme nadefinovali sťahovanie pošty vyššie spomínaného príkladu.

Na príkazovom riadku zadáme:

```
[mior@doma mior] $ crontab -e
```

(Všimnime si rozdiel v znaku promptu. Znak **#** symbolizuje konzolu roota, znak **\$** predstavuje konzolu ostatných používateľov Linuxu).

Po zadaní uvedeného príkazu sa spustí editor *vi*.

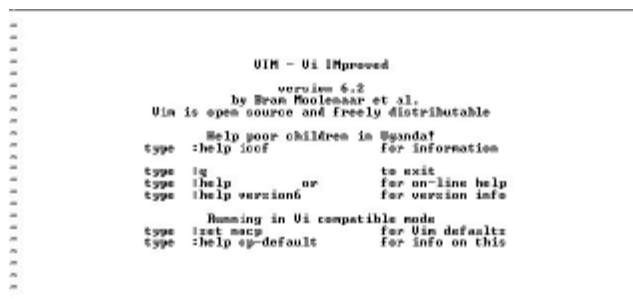
Čoooo?, Že vy neviete používať editor *vi*? To je ale chyba, pánové...

Mikrokurz editoru vi

Editor vi je najpoužívanejší editor v unixových systémoch a v Linuxe. Niekedy sa tradovalo, že kto nevie vi (slengovo pomenované víčko), ten si nesmie za unix (Linux) ani sadnúť. Dnes to už nie je pravda, tak ako neplatí, že kto nemá rozum, nech do politiky nelezie...

V poslednej dobe sa používa modernejší variant s názvom vim. Spúšťa sa ale rovnako ako jeho historický predchodca, jednoducho príkazom vi.

Zobrazí sa okno editora, ako je na obrázku č.2:



Tento editor nemá žiadne menu, žiadne makrá ani žiadne iné vymoženosti klasického editora. Ovláda sa pomocou kláves.

Ak chceme po spustení editora vi začať písať, musíme stlačiť kláves INSERT. Nachádza sa nad kurzorovými šípkami. Už názov insert (angl. vlož) značí vkladanie, v našom prípade textu.

Objaví sa kurzor a my napíšeme žiadaný text.

Po ukončení vkladania textu stlačíme kláves Esc. Tým opustíme editačný režim a prejdeme do režimu príkazového. Zadáme : (dvojbodku), ktorá sa objaví v ľavom dolnom rohu okna. Za dvojbodku môžeme zadať tieto príkazy:

- Ø qa! – na opustenie editora bez uloženia textu a zmien
- Ø wq – uloženie napísaného textu
- Ø help – na výpis helpu o editore vi

Ak chceme v editore vi editovať konkrétny súbor, zadáme príkaz vi meno_súboru.

Ešte jednoduchšie!

Možno je odo mňa kruté chcieť po vás ovládanie editora vi, keď dnes existujú iné a jednoduchšie editory.

Ja napríklad veľmi často používam editor, ktorý je súčasťou *Midnight Commandera*. To je ten editor, ktorý sa spustí po stlačení klávesu F4 v prostredí mc. Volá sa mcedit. (Ak to niekomu pripomína editor ncedit, ktorý je súčasťou *Norton Commandera*, tak má pravdu – nostalgia sa nezaprie.)

Poznám ho dobre a preto by som ho chcel používať všade v Linuxe, kde sa volá externý editor.

(Ak vy máte obľúbený iný editor, nevadí, tu uvedený postup platí všeobecne).

Ako ale donútiť program crontab, aby používal editor mcedit?

Už sme si vyššie spomenuli, že typ editoru je nadefinovaný v premennej systému s názvom EDITOR.

Najprv sa presvedčíme, aký editor je nadefinovaný.

V príkazovom riadku zadáme príkaz

```
[mior@doma mior] $ echo $EDITOR
```

Príkaz echo vypíše obsah premennej EDITOR.

Ak je táto premenná prázdna, je štandardne navolený editor vi, inak sa vypíše meno editora.

Teraz pretypujeme premennú EDITOR na hodnotu mcedit:

```
[mior@doma mior] $ export EDITOR=mcedit
```

Že sa tak naozaj stalo, môžeme si overiť už spomínaným príkazom echo.

Odteraz, keď spustíme program crontab, na jeho editáciu sa spustí editor mcedit. No a v tom sa pracuje podstatne príjemnejšie.

Poznámka:

Pretypovanie premennej z príkazového riadku je iba dočasné – do vypnutia systému. Ak chceme, aby sa premenná pretypovala pri každom štarte systému, musíme uložiť daný príkaz do štartovacích súborov. My už vieme, kde a ako!

Vráťme sa ale k vytvoreniu cron tabuľky používateľa mior:

Po zadaní príkazu `crontab -e` sa spustí náš obľúbený editor, ktorý vytvorí súbor `mior`. Teraz môžeme do súboru vkladať príkazy pre cron. V používateľských tabuľkách nie je nutné zadávať prvú časť súboru, ako je `PATH`, `MAILTO` a `HOME`, prejdeme rovno na časové údaje

```
* 6-18/2,22 * * * root /usr/bin/posta
```

Ukončíme editáciu a súbor uložíme. Takto sme vytvorili našu cron tabuľku. Nesmieme zabudnúť, že k súboru `posta` musíme mať prístupové práva, teda práva používateľa `mior`, inak cron zistí, že má čosi vykonať, ale to nevykoná, lebo nemá príslušné práva. Toto je veľmi častá chyba začiatočníkov (ale aj guru...).

Ak chceme len vypísať obsah cron tabuľky daného používateľa, zadáme príkaz `crontab -l`. V prípade, že chceme existujúci súbor zmazať, zadáme príkaz `crontab -r`.

Systémové cron adresáre

Všimnime si ešte raz obsah súboru `/etc/crontab` na výpise č.1 a pozrime sa bližšie napríklad na 1.časový riadok:

```
01 * * * * root run-parts /etc/cron.hourly
```

Povedzme si pozorne, čo tento zápis znamená:

V prvú minútu každej hodiny každého dňa sa pod právami roota spustí príkaz `run-parts` s parametrom `/etc/cron.hourly`.

Čo to znamená?

Cron má okrem súboru `/etc/crontab` a používateľských cron tabuliek ešte aj špeciálne 4 podadresáre v adresári `/etc/`: `cron.hourly`, `cron.daily`, `cron.weekly` a `cron.monthly`.

Sú to špeciálne adresáre, do ktorých v daný čas, určený v súbore `/etc/crontab`, nazrie program `run-parts` a ak nájde v danom adresári nejaký súbor – skript, vykoná ho.

Pozrime sa pre ilustráciu na adresár `/etc/cron.daily`.

Medzi inými obsahuje aj súbor `logrotate`. Jeho obsah je na výpise č.3:

```
#!/bin/sh
/usr/sbin/logrotate /etc/logrotate.conf
```

Vidíme, že sa jedná o spustiteľný skript, ktorého úlohou je následne spustiť príkaz `/usr/sbin/logrotate` s parametrom `/etc/logrotate.conf`.

Všimnime si, že vo všetkých štyroch podadresároch sa nachádzajú skripty, ktoré majú čo do činenia so systémom. A všimime si, že časové údaje u jednotlivých odkazov na príslušné podadresáre naozaj zodpovedajú vhodným názvom adresárov – každú hodinu, denne, týždenne a mesačne.

To však neznamená, že ich nemôžeme použiť pre naše účely. Stačí, ak do príslušného adresára vložíme skript s volaním žiadanej úlohy.

Obmedzovanie tvorby cron tabuliek

Cron má v sebe zabudovanú funkciu, že môže pomocou špeciálnych súborov rozhodnúť, kto môže a kto nemôže vytvárať cron tabuľky. Tie špeciálne súbory sú `/etc/cron.deny` a `/etc/cron.allow`.

Plnia tú istú úlohu ako súbory `allow-deny` iných démonov, napríklad `hosts`.

Princíp činnosti je tento:

Ak chceme niekomu zakázať používanie cronu, jeho meno zapíšeme do súboru `/etc/cron.deny`. Ostatní neuvedení používateľa systému budú mať povolenie cron používať!

A naopak:

Ak chceme niekomu povoliť používanie cronu, uvidíme jeho logovacie meno do súboru `/etc/cron.allow`. Ostatní neuvedení používateľa systému nebudú mať povolenie cron používať! Použijeme iba ten súbor, ktorý je pre nás jednoduchší, teda ten, kde sa menej píše. Nepoužívajme obidva súbory naraz – nemá to zmysel! Ak tieto súbory vôbec neexistujú, neuplatňuje sa žiadne obmedzenie.

Výstup z démonu cron

My môžeme využiť cron aj na odosielanie mejlových správ. V princípe využijeme schopnosť cronu poslať svoj výstup do iného programu, štandardne do mejlového programu.

Zápis vo všeobecnosti bude vyzeráť takto:

príkaz_pre_cron/mail mejlová_adresa

Taktiež môžeme výstup presmerovať do súboru, napr. `cron.log` takto:

príkaz_pre_cron>>/var/log/cron.log

Vezmime si tento ilustratívny príklad:

Nech máme takúto cron tabuľku:

15 7 3 5 * echo Mama ma narodeniny mail mior@doma.sk
20 7 13 2 * echo Manzelka ma narodeniny mail mior@mobil.sk
25 17 * * * /usr/bin/zaloha>>/var/log/zaloha.log
20 21 * * * /usr/bin/kontrola>>/dev/null

Prečítajme si, čo táto tabuľka znamená:

Prvý riadok hovorí:

Každý rok 3.mája o 7,15 hodine mi na adresu mior@doma.sk príde mejl, ktorý bude obsahovať vetu „Mama ma narodeniny“. Ja si tento mejlík prečítam a mame zavolám. A keďže moja dobrá mama býva ďaleko a je už na dôchodku, je jedno, kedy si mejl prečítam, či hneď ráno alebo poobede.

Druhý riadok hovorí:

Každý rok 13.februára o 7,20 hodine cron odošle na adresu mior@mobil.sk mejl, ktorý bude obsahovať text „Manzelka ma narodeniny“. Tento mejl je však schránka u operátora mobilných telefónov a je to taká služba, že akonáhle príde správa na túto adresu, obsah správy mi daný operátor prepošle ako SMS - správu na môj mobilný telefón. Nie som závislý od toho, kedy si mejl prečítam, lebo mobil mám (spravidla) so sebou a po prijatí SMS správy okamžite kupujem obrovskú kyticu ruží a diamantový náhrdelník a odchádzam za svojou milovanou polovičkou.

Tretí riadok:

Tento riadok už poznáme. Každý deň o 17,25 hodine sa vykoná skript s názvom *záloha* a jeho výstupné hlášky sa uložia do súboru `/var/log/zaloha.log`. A tam si ich nájdem a vo vhodnú chvíľu preštudujem, či všetko prebehlo v poriadku.

Štvrtý riadok:

Každý deň o 21,20 hodine sa spustí skript `/usr/bin/kontrola` a jeho textové výstupy sa pošlú na zariadenie `/dev/null`. Toto je špeciálne zariadenie v Linuxe a môžeme si pod ním predstaviť bezodný odpadkový kôš. Všetko, čo pošleme na toto zariadenie sa navždy zahodí (a nemožno to obnoviť!).

Program at

Program *at* je podstatne jednoduchší ako príkaz *cron*. Nemá ani žiadne konfiguračné súbory ani žiadne tabuľky, všetko sa zadáva z príkazového riadku. Jeho úlohou je spustiť konkrétnu činnosť v zadanom čase. Keď sa daná úloha splní, príkaz *at* sa ukončí.

Všeobecné zadanie príkazu *at* je

at časový_údaj

Hodnoty pre program at

Časový údaj je parametrom programu *at* a môže dosahovať týchto hodnôt:

- Ø špecifikácia času vo forme HH:MM
- Ø špecifikácia dátumu vo forme
 - o MMDDRR
 - o MM/DD/RR
 - o DD.MM.RR

Ø časová jednotka v tvare *at čas + časova_jednotka* vo forme

- minutes
- hours
- days
- weeks
- tomorrow
- today

Ø presný čas vo forme

- noon
- midnight
- teatime (t.j. o 16,00 hod)
- now

napríklad *at 15:00 + 5 days* značí, že sa daná úloha vykoná za 5 dní o tretej poobede.

Ak pridávame aj dátum, dávame ho zásadne za časový údaj, napr. *at 17:30 122404*, čo bude na štedrý deň o pol šiestej večer.

Ale kde zadáme príkaz, ktorý chceme vykonať?

Po zadaní príkazu *at* s daným časom sa na obrazovke objaví nový prompt so znakom zobáčika >. Na riadok s týmto promptom zadávame príkazy, ktoré sa majú vykonať v danom čase. Po zadaní posledného príkazu a potvrdení klávesom Enter stlačíme kombináciu klávesov Ctrl-D. Tým sa uzatvorí editovanie programu *at* a žiadané príkazy sa ako úloha uložia do fronty a čakajú na ten správny čas, kedy sa vykonajú.

Ovládanie fronty *at*

Na ovládanie fronty úloh, ktoré čakajú na spustenie, existujú tieto príkazy:

- Ø *atq*
- Ø *at -c*
- Ø *atrm*

Príkaz **atq** vypíše existujúce úlohy, čakajúce vo fronte na vykonanie.

Zadáme príkaz

```
[root@doma root] # atq
```

a na obrazovke sa zobrazí, napr.:

1	2004-11-10 14:51 a root
---	-------------------------

Vypíše sa identifikačné číslo úlohy, dátum a čas vykonania, typ úlohy a vlastníka úlohy.

Typy úlohy sú:

- Ø **a** je štandardný typ
- Ø **b** je pre úlohu patriacu príkazu *batch* (nebudeme dnes preberať)

Ak chceme vidieť obsah úlohy, ktorá sa v určitú dobu vykoná, použijeme príkaz

at -c identifikačné_číslo_úlohy,

napríklad

```
[root@doma root] # at -c 1
```

a na obrazovke sa objaví výpis príkazov, tak ako sme ho pre danú úlohu zadali.

Príkaz **atrm** odstráni danú úlohu z fronty. Syntax príkazu je *atrm identifikačné_číslo_úlohy*.

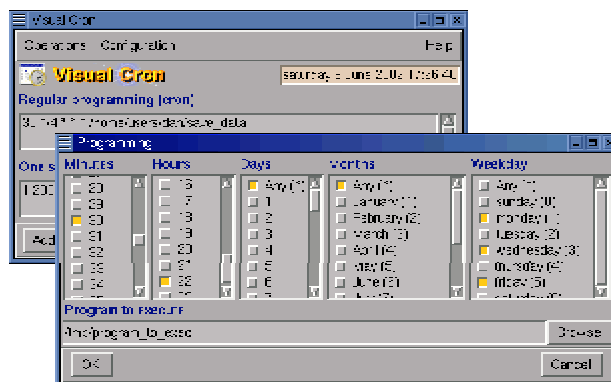
Tak ako *cron*, tak aj program *at* môže disponovať súbormi *at.allow* alebo *at.deny* v adreári */etc/*.

Ak má mať užívateľ možnosť používať program *at*, musí byť uvedený v súbore */etc/at.allow*, resp. nesmie sa nachádzať v súbore */etc/at.deny*. Pokiaľ neexistuje ani jeden z týchto súborov, nikto okrem roota nemá možnosť disponovať s programom *at*. Ak existuje len */etc/at.deny* a je prázdny, potom môže s *at* pracovať každý.

Vizuálne nástroje

Aby sme potešili aj desktopistov, ktorí nemajú príkazový riadok veľmi v láske, existuje dostatočné množstvo vizuálnych nástrojov na správu programov *cron* a *at* v grafickom prostredí. Jeden z nich je aj program *vcron* francúza Daniela Roché, ktorý získame na stránke <http://www.linux-kheops.com/pub/vcron/vcronGB.html>.

Screenshot nadstavby *vcron* je na obrázku č.4:



Záverom...

Touto spoločnou časťou pre serveristov aj desktopistov sa s desktopistami čiastočne lúčim.

V sekcii desktop sme prebrali (skoro) všetko, čo by mal desktopista v podstate vedieť.

Hovorím čiastočne, lebo keď sa v desktope objaví niečo nové (a to sa isto objaví...), povieme si o tom pár slov.

A v prípade, že by mali desktopisti pocit, že sme sa ešte nenaučili to či ono, nech mi zamejľujú na mior@psg.sk a zase sa v sekcii desktopu s konkrétnou témou stretneme.

Desktopisti však nemusia byť smutní, že sú opúšťaní. Niekde v týchto miestach začína nový seriál o OpenGL pod Linuxom (čo je o grafike) od môjho kolegu Mareka Sopka.

(My) serveristi sa naďalej budeme venovať serverom, tu nás čaká ešte poriadny kus práce.

A okrem iného – aj serveristi aj desktopisti - sa pustíme do nového, už avizovaného a mnohými očakávaného seriálu **Linux extra**, kde budeme k Linuxu pripájať rôzne zariadenia. Začneme jednoduchými svetelnými diódami, tlačidlami, prejdeme na rôzne reléové spínače a budeme sa bavkať aj viacriadkovými LCD displejmi. Čo to počujem, že to nie je vôbec pre desktopistov? Ale ako hovorí český klasik – „Prdlavka, pánové....“. Ešte som nevidel desktopistu, ktorý by nechcel mať k svojmu počítaču pripojený mobil!

Linux prakticky ako server / 17.časť

Aj u vás sa to stáva? Celá sieť, servery aj klientské počítače si tŕško bručia a pracujú na požadovaný výkon. Keďže sa vonku celkom zozimilo a v kanceláriách sa kúri nedostatočne, milá sladučká kolegynka s copíkmi na hlave si priniesla z domu šikovný ohrievač. Reku, že si troška nôžky pozohrieva. No a akože inak, v plnej odberovej špičke pripojí dvojkilowattovú špirálku do energetickej sústavy energetických závodov. A keďže sú dnes skoro všade namontované strážiče odberových špičiek, dôjde ku krátkodobému výpadku elektriny. Čo dokáže urobiť aj len niekoľko sekundový výpadok elektrického prúdu s počítačmi, to si nemusíme hovoriť. Keď sa „seknú“ klientske stanice, to sa dá ešte celkom prežiť, za predpokladu, že si pravidelne ukladáme dôležité súbory na iný (sieťový) disk alebo médium. Keď sa však „dobré“ sekne server, tak aj pri najlepšej vôli, všetkých dostupných zálohách a úprimnej snahe môže jeho výpadok trvať hodiny. A v zmysle porekadla, že „čas sú peniaze“, niekoľkohodinový výpadok môže mať naozaj značné ekonomické následky. My už nie sme v oblasti IT začiatočníci a tak vieme, že krátkodobé výpadky sa dajú celkom úspešne eliminovať pomocou tzv. UPS, ktoré bežne dokážeme k počítaču s operačným systémom MS Windows pripojiť. No hej, ale ako to zariadiť, aby fungovali pod Linuxom? A o tom sa dnes budeme rozprávať.

Teória o UPS

UPS – (*Uninterruptible Power Supply*) – zdroj neprerušovaného napájania je vo svojej podstate prístroj, ktorý v prípade výpadku elektrickej energie z rozvodnej siete okamžite začne dodávať pripojenému zariadeniu náhradné striedavé napätie 220V. Toto napätie vyrobí vstavaná elektronika a zdrojom energie je akumulátor. Takto vyrobená energia sa dodáva do vyčerpania akumulátora alebo do obnovenia dodávky elektriny z rozvodnej siete. Po obnove dodávky sa napájaný počítač zapne, systém sa spustí a v UPS sa vyčerpaný akumulátor dobije na požadovanú hodnotu. Nič zložitého!

Maximálna doba, po ktorú je UPS schopná vyrábať elektrickú energiu závisí od jej kapacity, zaťaženia a stavu akumulátorov. Spravidla to býva v rozmedzí od 10 do 30 minút.

Existujú však chytré UPS-ky, ktoré dokážu nielen vyrobiť elektrinu, ale aj napájanému počítaču povedať, že došlo k prerušeniu dodávky energie z rozvodu a v prípade, že sa akumulátor začína nebezpečne vyčerpávať, povie počítaču, aby sa korektne vypol, aby nedošlo k strate údajov.

Aby mohla UPS-ka počítač informovať o prerušení alebo obnovení dodávky energie, musí byť medzi UPS a počítačom vytvorený komunikačný spoj. Ten sa spravidla realizuje pomocou špeciálneho káblu, ktorý vychádza z UPS a na počítač sa pripája pomocou sériového alebo USB portu.

Prípojný kábel býva súčasťou dodávky zdroja. Ak tak nie je, je možné správne pripojenie káblu nájsť na Internete.

Rozdelenie UPS

UPS môžeme rozdeliť podľa viacerých hľadísk:

- Ø podľa určenia
- Ø podľa montáže
- Ø podľa inteligencie

Podľa určenia rozdeľujeme UPS na zdroje pre pracovné stanice, pre servery, pre dátové centrá, pre jednosmerné zálohovanie, prenosné zdroje a podobne.

Podľa montáže rozdeľujeme zdroje na stolné (*standalone*) (obr.č.1),



montovateľné do racku (obr.č.2)



a stojanové (obr.č.3):



Rozdelenie podľa inteligencie

Ako sme si povedali vyššie, existujú jednoduché zdroje a chytré zdroje. Každý výrobca UPS jednotlivé kategórie zdrojov rozlišuje názvami. Ako príklad si môžeme zobrať firmu *American Power Conversion* (APC).

Obyčajné UPS sú označované **Back UPS**. Tie chytré sú označované – akože ináč – **Smart UPS** (smart = chytrý, bystrý).

Spoznáme ich podľa konektora na zadnej strane zariadenia – obr.č.4:



Ovládací softvér

Aby sme mohli počítač ovládať, musíme mať na príslušnom počítači nainštalovaný príslušný softvér. Keďže na priloženom CD k UPS nájdeme (spravidla iba) softvér pre operačný systém MS Windows, musíme si soft pre náš server stiahnuť z Internetu.

Medzi Linuxákmi sa rozšírili tieto tri typy softvéru:

- Ø PowerChute
- Ø apcupsd
- Ø NUT

PowerChute

PowerChute je program, ktorý vytvorila firma APC pre ovládanie svojich UPS. V minulosti bol tento produkt komerčný aj pre Linux. Neskôr firma zmenila politiku a dnes je tento softvér voľne stiahnuteľný z ich webovej stránky www.apcc.com. Bohužiaľ, nedodáva sa vrátane zdrojových kódov, ale len ako binárka a preto sa môže stať, že pod niektorou distribúciou Linuxu nepôjde nainštalovať alebo nebude korektne fungovať.

apcupsd

Práve v minulosti voľná nedostupnosť programu *PowerChute* prinútila linuxových vývojárov vytvoriť podobný program. A tak vznikol *apcupsd*.

Ako jeho názov naznačuje, je určený iba pre zdroje od firmy APC. Ale keďže firma APC má najväčší podiel na trhu s UPS, je veľmi hojne využívaný.

NUT

Obmedzenie programu *apcupsd* pre zdroje od firmy APC podnietilo vznik programu *NUT (Network UPS Tools)*. Ten podporuje mnoho UPS od rôznych výrobcov, napr. APC, Belkin, Best Power, MGE UPS Systems, Oneac, Online, PowerTech, Soltec, UPSonic, Victron a iné.

My sa budeme venovať sprevádzkovaniu programu *apcupsd* a *NUT*.

Podstata činnosti apcupsd

Domovská stránka projektu *apcupsd* je na www.apcupsd.com. Môžeme tu nájsť program ako pre Linux, tak aj pre operačný systém MS Windows.

Pre Linux si môžeme vybrať zdrojové kódy v balíku tar.gz. alebo ako balík binárok vo formáte rpm pre najznámejšie distribúcie a ich verzie – RedHat, Fedora Core, SUSE a Mandrake.

Program *apcupsd* môžeme používať v troch režimoch:

- Ø samostatný (standalone) režim
- Ø sieťový režim
- Ø režim zdieľania UPS

Samostatný režim

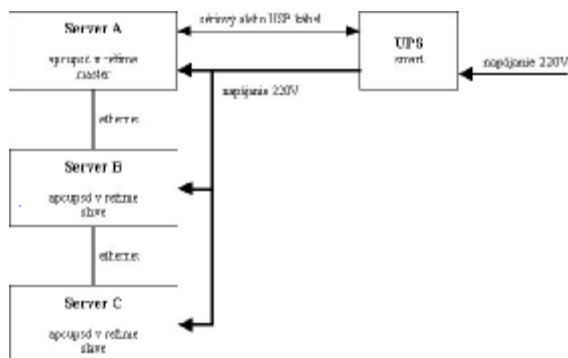
Princíp samostatného režimu je na obr.č.5:



Využíva sa vtedy, keď z jedného zdroja UPS napájame jeden počítač, v našom prípade server. Je to najčastejšie použitie zapojenia UPS.

Sieťový režim

Princíp sieťového režimu je na obr.č.6:



Využíva sa vtedy, ak z jedného zdroja UPS napájame viac serverov. My však chceme, aby sa všetky servery korektne vypli. Vtedy je na serveri, ktorý pomocou sériového alebo USB káblu komunikuje so zdrojom UPS, program *apcupsd* v režime **master**. Ostatné servery sú medzi sebou prepojené počítačovou sieťou a program *apcupsd* je v režime **slave**. Keď nastane výpadok elektrickej energie, UPS pošle signál serveru A. Ten to oznámi cez ethernetovú sieť ostatným serverom B a C.

Režim zdieľania UPS

Princíp tohto režimu je na obr.č.7:



Podstata spočíva vo využívaní viacerých zdrojov UPS medzi viaceré počítače. Využíva sa hlavne v serverových farmách.

My si teraz sprevádzkujeme standalone režim.

Predstavme si, že používame **APC SmartUPS 420E** a používame distribúciu Fedora Core 1, a tak si stiahneme balíček *apcupsd-std-3.10.16-1.i386.fc1.rpm*.

Po jeho nainštalovaní sa v adresári */etc/* vytvorí nový podadresár *apcupsd*. V ňom sa nachádza konfiguračný súbor *apcupsd.conf*.

Spustíme ho v príslušnom editore a v ňom nastavíme tieto parametre:

- Ø UPSCABLE na UPSCABLE smart {určuje typ komunikačného káblu}
- Ø UPSTYPE na UPSTYPE smartups {určuje typ inteligencie UPS}
- Ø DEVICE na DEVICE /dev/ttyS0 {určuje port pripojenia – ttySx pre sériový, hiddev[0-15] pre USB port}
- Ø BATTERYLEVEL na BATTERYLEVEL 5 {udáva kapacitu v %, ak pod túto hodnotu klesne stav akumulátorov, UPS vyšle signál shutdown}
- Ø MINUTES na MINUTES 3 {UPS dokáže odhadnúť svoju výdrž. Ak jej zostáva už len zadaná hodnota – 3 minúty, tak vyšle signál shutdown}
- Ø NETSERVER na NETSERVER off {"on" len v prípade sieťového režimu}

Súbor upravíme a uložíme.

Službu spustíme známym príkazom **service apcupsd start**.

Odteraz máme monitorovaný zdroj UPS.

Preverka zdroja UPS

O tom, či je zdroj UPS naozaj monitorovaný a v akom je stave sa môžeme presvedčiť viacerými spôsobmi. Ak si to chceme overiť z konzoly, na príkazovom riadku zadáme príkaz

```
[root@rubin lib] # apcaccess
```

Na obrazovke počítača sa objaví textový výpis stavu UPS – výpis č.8:

```
APC      : 001,049,1180
DATE     : Sun Dec 05 12:25:02 CET 2004
HOSTNAME : rubin.doma.sk
RELEASE  : 3.10.16
VERSION  : 3.10.16 (04 November 2004) redhat
UPSNAME  : UPS_IDEN
CABLE    : Custom Cable Smart
MODEL    : Smart-UPS 420
UPSMODE  : Stand Alone
STARTTIME: Sun Dec 05 08:05:07 CET 2004
STATUS   : ONLINE
LINEV    : 214.5 Volts
LOADPCT  : 42.2 Percent Load Capacity
BCHARGE  : 100.0 Percent
TIMELEFT : 18.0 Minutes
MBATTCHG : 5 Percent
MINTIMEL : 3 Minutes
.
.(skrátene)
.
ALARMDEL : 5 seconds
BATTV    : 13.8 Volts
LINEFREQ : 50.0 Hz
.
.(skrátene)
.
MANDATE  : 11/04/01
SERIALNO : NS0144351829
BATTDATA : 11/04/01
NOMOUTV  : 230
NOMBATTV : 12.0
FIRMWARE : 21.6.I
APCMODEL : DWI
END APC  : Sun Dec 05 12:25:48 CET 2004
```

Stav UPS môžeme kontrolovať aj na diaľku pomocou grafického rozhrania. To je napísané tak, že využíva http server, takže na prehliadanie stačí obyčajný internetový prehliadač.

V adresári `/etc/apcupsd/` sa nachádza aj podadresár `/cgi/`. Obsahuje niekoľko súborov, ktoré zabezpečujú tvorbu grafického rozhrania. Tieto súbory presunieme do adresára http servera, kde sa ukladajú spustiteľné skripty.

V distribúcii FC1 je to adresár `/var/www/cgi-bin/`. Najdôležitejší súbor je súbor s názvom ***multimon.cgi***.

Potom stačí spustiť web prehliadač, zadať adresu servera vrátane cesty k súboru *multimon.cgi*, teda napríklad:

<http://192.168.10.1/cgi-bin/multimon.cgi>

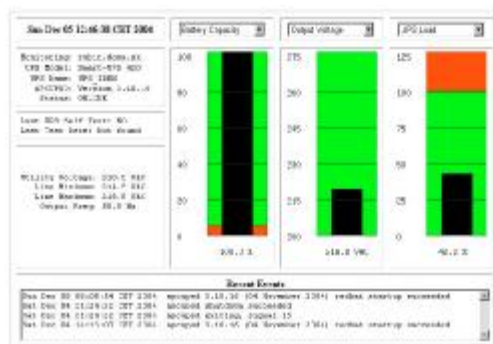
V okne prehliadača uvidíme stav UPS – obr.č.9:



Vidíme typ modelu, ktorý program vyčítal priamo zo zdroja UPS, **Status** (ONLINE), kapacitu akumulátorov (100%), približný čas výdrže (24 minút). Ak klikneme na posledné okienko **DATA** (All data), uvidíme výpis parametrov tak, ako na výpise č.8.

V prípade, že by boli v sieti aktívne viaceré zdroje UPS, bude tento APCUPS Network Monitor obsahovať viac riadkov.

Ak klikneme na prvé okienko **System** (Local Host), zobrazí sa nové okno s grafickým vyjadrením žiadaných hodnôt – obr.č.10:



Zobrazia sa tri stĺpcové grafy, ktorých význam si môžeme ľubovoľne zvoliť pomocou výberového okna nad každým grafom.

A odteraz môžeme pokojnejšie spať, lebo sme zase urobili niečo naviac k zabezpečeniu nášho servera.

Nabudúce sa budeme venovať programu *NUT* a jeho grafickým nastaveniam.

Linux prakticky ako server / 18.časť

V minulej časti sme si povedali niečo o zdrojoch nepretržitého napájania – tzv. UPS a zároveň sme si ukázali, ako sprevádzkovať program *apcupsd*. Ak sme postupovali presne podľa návodu, museli sme sa dopracovať k zdárnemu úspechu. Prečo? Lebo program *apcupsd* je relatívne jednoduchý. Má však jednu chybu, tak ako sme si ju spomenuli. Je šitý presne na mieru zariadeniam od firmy *American Power Conversion* (APC). Čo však v prípade, ak máme UPS-ku od iného výrobcu?

Vtedy nastupuje program **NUT**, o ktorom si dnes budeme rozprávať.

NUT

NUT je skratka z *Network UPS Tools*. Domovská stránka je na adrese <http://networkupstools.org>, kde je možné stiahnuť okrem zdrojových kódov aj inštalačné balíky pre rôzne distribúcie Linuxu, ako je RedHat 9, Fedora Core, Slackware, SUSE, Gontoo a podobne. Takže ak máme niektorú z uvedených distribúcií, máme inštaláciu podstatne zjednodušenú.

Na rozdiel od programu *apcupsd* je filozofia NUT trochu zložitejšia – obr.č.1:



Všimnime si, že na ceste k UPS ležia na strane Linuxu tri časti:

- Ø driver
- Ø server
- Ø klient

Driver

Driver – ovládač je program, ktorý sprostredkuje komunikáciu medzi fyzickým zariadením UPS a linuxovým počítačom. Pre každý typ UPS je potrebný iný ovládač.

Pre UPS od firmy APC je to ovládač s menom *apcsmart*, pre UPS od firmy Best Power je to ovládač *bestup*, pre UPS od firmy Victron je to ovládač *victronups* a podobne.

Meno konkrétneho ovládača pre tú našu UPS-ku nájdeme v dokumentácii alebo pohl'adáme na Internete.

Súčasťou inštalačného balíku sú ovládače pre najznámejšie typy UPS (asi 30 druhov).

Server

Tu pod serverom rozumieme démona, ktorý sprostredkuje komunikáciu medzi ovládačom a ostatnými programami - klientami. Je iba jeden a nazýva sa ***upsd***. Jeho úlohou je byť stále prítomný v systéme.

Klient

Klient je program, ktorý prijíma zo serveru *upsd* správy o stave UPS a vykonáva požadovanú činnosť, napr. korektne vypne počítač, nastavuje rôzne parametre UPS a podobne. Klientov môže byť niekoľko druhov a dokonca ich môže byť v systéme viac naraz.

Inštalácia

„Tak ako všetko v Linuxe“, tak aj inštalácia programu NUT sa dá vykonať rôznymi spôsobmi.

Inštalácia zo zdrojových kódov

Ešte predtým, ako začneme, vytvoríme v našom systéme používateľa **nut** a skupinu **nut**.

Potom si stiahneme z Internetu zdrojové kódy programu, napr. **nut-2.0.0.tar.gz**.

Uložíme ho do niektorého pomocného adresára, napr. */install/nut/*. Prejdeme do tohto adresára príkazom

```
[root@rubin lib] # cd /install/nut
```

Balíček rozbalíme príkazom:

```
[root@rubin nut] # tar xzfvp nut-2.0.0.tar.gz
```

Ztarovaný balík sa rozbalí a vytvorí nový podadresár *nut-2.0.0*.
Vstúpime do tohto podadresára príkazom

```
[root@rubin nut] # cd nut-2.0.0
```

Odteraz všetky kompilačné činnosti budeme robiť v tomto adresári.

Poznámka:

Musíme si zopakovať, aký je najklasickjší postup pri inštalácii programu zo zdrojových kódov. My vieme, že zdrojové kódy sú v podstate textové súbory, ktoré sú bežnými textovými editormi čitateľné a upravovateľné. Obsahujú spravidla príkazy programovacieho nástroja, čo najčastejšie býva na 99,99% jazyk C (môžu to byť aj iné jazyky, ako pascal, python atď., ale tie majú iný postup). Aby tieto príkazy mohli byť počítačom vykonateľné, musíme mu ich preložiť do strojového jazyka.

Preloženie pozostáva z viacerých krokov (nebojme sa, nemusíme byť programátory v Cečku!) .

V prvom rade musíme vykonať predprípravu na kompiláciu – konfiguráciu zdrojových kódov. Pri konfigurácii môžeme zadávať rôzne parametre, ktorými chceme dosiahnuť žiadaný efekt pri kompilácii. Následným krokom je samotná kompilácia, po ktorej nasleduje postkompilačné prekopírovanie vytvorených súborov do príslušných adresárov. Veď si to teraz prakticky ukážeme:

Začneme konfiguráciou. V danom adresári *nut-2.0.0* zadáme príkaz

```
[root@rubin nut-2.0.0] # ./configure --with-user=nut
```

Poznámka:

Aby nedošlo k interpretačnej chybe, foneticky zápis je:

bodka lomka configure medzera mínus mínus with mínus user rovná sa nut.

Zvlášť upozorňujem na tie dve mínusky pred slovom „with“!

Parametrom **with-user** sme nadefinovali, že sa pri kompilácii vytvoria súbory, ktorých vlastník je nový používateľ Linuxu s menom **nut**. Na obrazovke prebehne dlhý text výpisov, ktoré ani nestihneme sledovať, ale to si teraz nevšímajme.

Až skončí príkaz *configure*, pristúpime k samotnej kompilácii príkazom

```
[root@rubin nut-2.0.0] # make
```

Tým sa vytvoria binárne súbory. Teraz ich príkazom

```
[root@rubin nut-2.0.0] # make install
```

presunieme do požadovaných adresárov, v tomto prípade do adresára */usr/local/ups/*.

Takto sme vykonali preloženie zdrojových kódov. Aby sme získali príklady konfiguračných súborov, zadáme príkaz

```
[root@rubin nut-2.0.0] # make install-conf
```

Pri tejto inštalácii sa nevytvoria CGI skripty. Sú to skripty, ktoré môžeme volať pomocou internetového prehliadača a výsledok uvidíme v prehliadači.

Aby sme dosiahli vytvorenie CGI skriptov, zadáme príkazy

```
[root@rubin nut-2.0.0] # ./configure --with-user=nut --with-cgi  
[root@rubin nut-2.0.0] # make cgi  
[root@rubin nut-2.0.0] # make install-cgi  
[root@rubin nut-2.0.0] # make install-cgi-conf
```

UPS na USB

Môže sa stať, že máme UPS-ku, ktorá na komunikáciu s počítačom používa USB port.

Pre takúto komunikáciu potrebujeme súbor **hiddev.h**, ktorý sa najčastejšie nachádza v adresári `/usr/include/linux`.

Vtedy zadáme príkazy:

```
[root@rubin nut-2.0.0] # ./configure --with-linux-hiddev=/usr/include/linux/hiddev.h
[root@rubin nut-2.0.0] # make usb
[root@rubin nut-2.0.0] # make install-usb
```

Ak nepoužívame UPS pripojenú na USB, tak predchádzajúci odsek preskočíme.

Po vykonaní inštalácie musíme ešte urobiť pár úkonov, ako je vytvorenie špeciálneho adresára, priradenie vlastníkov a práv a úprava vlastníctva sériového portu.

Takže vytvoríme adresár `/var/state/ups`:

```
[root@rubin nut-2.0.0] # mkdir -p /var/state/ups
```

Potom upravíme prístupové práva:

```
[root@rubin nut-2.0.0] # chmod 0700 /var/state/ups
```

a nakoniec tento adresár priradíme vlastníkovi:

```
[root@rubin nut-2.0.0] # chown nut:nut /var/state/ups
```

Teraz upravíme práva a vlastníctvo k portu, na ktorom je pripojená UPS. Nech je to v našom prípade prvý sériový port – **ttyS0**.

Zmeníme práva príkazom

```
[root@rubin nut-2.0.0] # chmod 0600 /dev/ttyS0
```

a zmeníme vlastníka

```
[root@rubin nut-2.0.0] # chown nut:nut /dev/ttyS0
```

To by bolo k samotnej inštalácii zo zdrojových kódov všetko.

Inštalácia z balíčkov

Ak vykonávame inštaláciu z príslušného balíčka vhodného pre našu distribúciu, vyššie spomínané úkony budú za nás prevedené balíčkovacím nástrojom. Použitie príslušného balíčkovacieho programu, napr. RPM sme si už ukazovali viackrát.

Ostatné tu nižšie popísané úkony sú spoločné pre všetky distribúcie a spôsoby inštalácie.

Dôležité súbory

Po inštalácii vzniknú tieto dôležité súbory:

- Ø **upsmon** – monitoruje stav UPS a v prípade potreby vyvoláva shutdown. Na základe stavu UPS spúšťa predom definované akcie či skripty
- Ø **upsd** - démon
- Ø **upslog** – pravidelne zapisuje stav UPS do súboru
- Ø **upsc** – utilita pre príkazový riadok – vypisuje aktuálny stav UPS
- Ø **upsw** – nástroj na úpravu parametrov UPS
- Ø **upsstats.cgi** – webové rozhranie zobrazujúce stav UPS – podobne ako u **apcupsd**
- Ø **upsset.cgi** – webové rozhranie na nastavovanie UPS
- Ø konfiguračné súbory

Nastavenie

Po samotnej inštalácii prejdeme k nastaveniu jednotlivých konfiguračných súborov:

ups.conf

V prvom rade upravíme súbor **ups.conf**. Jeho umiestnenie závisí od spôsobu inštalácie, ale spravidla sa nachádza v adresári `/etc/ups/` alebo `/usr/local/upc/etc/`.

Jeho obsahom je stanovenie príslušného ovládača pre danú UPS-ku. Keďže som majiteľom UPS od firmy APC a to typu Smart 420, použijem driver *apcsmart*. UPS je pripojená na porte `ttyS0` – (výpis.č.2):

```
[smart420]          # meno UPS
    driver = apcsmart # ovladač
    port = /dev/ttyS0 # port
```

upsd.conf

Ďalší súbor **upsd.conf** obsahuje definície takzvaných zoznamov prístupu riadenia. Tým sa hovorí po anglicky *Access Control List* – ACL.

Najzákladnejšie nastavenie súboru **upsd.conf** je na výpise č.3:

```
ACL all 0.0.0.0/0
ACL localhost 127.0.0.1/32

ACCEPT localhost
REJECT all
```

upsd.users

Teraz nastavíme súbor **upsd.users**.

Nadefinujeme v ňom používateľa **monuser**, ktorému pridáme heslo **test**, bude pochádzať z **localhostu** a bude typu **master**.

Výpis súboru **upsd.users** je na výpise č.4:

```
[monuser]
    password = test
    allowfrom = localhost
    upsmon master
```

upsmon.conf

Nakoniec pristúpime ku konfigurácii súboru **upsmon.conf**.

Ten obsahuje mnoho parametrov, my si ukážeme najdôležitejšie z nich – výpis č.5:

```
# upsmon.conf
#
# záložný zdroj, ktorý budeme monitorovať
MONITOR smart420@localhost 1 monuser test master
#
# počet záložných zdrojov, ktoré sú v prevádzke
MINSUPPLIES 1
#
# príkaz pre spustenie shutdown príkazu
SHUTDOWNCMD "/sbin/shutdown -h +0"
#
# ako často upsmon kontroluje stav UPS (v sekundách)
POLLFREQ 5
#
# ako často upsmon kontroluje stav UPS v prípade,
# že je prepnutá na napájanie z akumulátoru (v sekundách)
POLLFREQALERT 5
#
# cesta, kde upsmon založí súbor, ktorého existenciu
# testujeme v záverečnej fáze shutdown sekvencie a v prípade,
# že súbor existuje, dáme UPS príkaz k vypnutiu
```

```
POWERDOWNFLAG /etc/killpower
```

Spustenie NUT

Pri testovaní konfigurácie NUT vykonáme jeho spustenie po jednotlivých krokoch, aby sme odladili prípadné chyby.

V prvom rade zaktivujeme driver. Zadáme príkaz

```
[root@rubin root] # upsdrcvctl start
```

Na obrazovke sa ukáže výpis č.6:

```
Network UPS Tools – UPS driver controller 2.0.0
Network UPS Tools (version 2.0.0) – APC Smart protocol driver
  Driver version 1.99.6, command table version 2.0
Detected Smart-UPS 420  [NS0144351829] on /dev/ttyS0
```

Po spustení ovládača môžeme spustiť samotný UPS server príkazom

```
[root@rubin root] # upsd
```

Na obrazovke sa ukáže výpis č.7:

```
Network UPS Tools upsd 2.0.0
/var/state/ups is world readable
/etc/ups/upsd.conf is world readable
Connected to UPS [smart420]: apcsmart-ttyS0
Synchronizing...done
```

Teraz nastáva chvíľa k spusteniu klientov.

Či je spojenie s UPS aktívne a funkčné, overíme pomocou vyššie spomenutého príkazu **upsc**. Ako parameter zadáme názov UPS, definovaný v súbore *ups.conf*:

```
[root@rubin root] # upsc smart420@localhost
```

Na obrazovku vyroluje dlhší výpis stavu UPS – výpis č.8:

```
battery.alarm.threshold: 0
battery.charge: 100.0
battery.charge.restart: 00
battery.date: 11/04/01
battery.runtime: 1500
battery.runtime.low: 120
battery.voltage: 13.85
battery.voltage.nominal: 012
driver.name: apcsmart
driver.parameter.port: /dev/ttyS0
driver.version: 2.0.0
driver.version.internal: 1.99.6
input.frequency: 50.00
input.quality: FF
input.sensitivity: M
input.transfer.high: 253
input.transfer.low: 208
input.transfer.reason: R
input.voltage: 217.4
input.voltage.maximum: 221.7
input.voltage.minimum: 218.8
output.voltage: 218.8
output.voltage.target.battery: 230
ups.delay.shutdown: 020
```

```
ups.delay.start: 000
ups.firmware: 21.6.I
ups.id: UPS_IDEN
ups.load: 041.6
ups.mfr: APC
ups.mfr.date: 11/04/01
ups.model: Smart-UPS 420
ups.serial: NS0144351829
ups.status: OL
ups.test.interval: 0
ups.test.result: NO
```

Teraz môžeme spustiť ľubovoľného klienta, napríklad **upsmon** príkazom:

```
[root@rubin root] # upsmon start
```

Jeho štart je oznámený hláškou na výpise č.9:

```
Network UPS Tools upsmon 2.0.0
UPS: smart420@localhost (master) (power value 1)
Using power down flag file /etc/killpower
```

Takto máme zabezpečený server pomocou UPS.

Ak všetko funguje ako má, vytvoríme príslušné spúšťacie skripty.

V prípade, že sme inštalovali z balíčkov, môžeme očakávať, že spúšťacie skripty boli automaticky vytvorené a uložené do jednotlivých run-levelov. Vtedy stačí, aby sme použili programy na obsluhu služieb, napr. v distribúcii RedHat alebo Fedora môžeme použiť známy príkaz **service ups start|stop|restart**.

Grafickí klienti

Môže sa stať, že o stave UPS chceme byť informovaní cez grafické rozhranie. (To preto, že tento článok si môžu prečítať a naštudovať aj desktopisti).

Vtedy môžeme použiť CGI skripty alebo grafických klientov.

Pre prostredie KDE je najznámejší klient **KNutClient** od českého autora Daniela Prynychy.

KNutClient

Tohto klienta môžeme nainštalovať zo zdrojových kódov alebo z distribučného balíčka. Po inštalácii musíme vykonať nastavenie.

Klikneme na *New UPS* a na karte vyplníme požadované údaje – obr.č.10:



Zvolíme vhodné meno, ako **UPS address** zadáme meno servera, ku ktorému je UPS pripojená (*localhost*) a ako **UPS name** zadáme meno, ktoré sme nadefinovali v súbore *ups.conf* (*smart420*). V dolnej časti okna vyberieme, ktoré parametre UPS-ky budeme sledovať.

Na záložke **Setting** definujeme, v koľkých stĺpcoch sa budú parametre zobrazovať – obr.č.11:



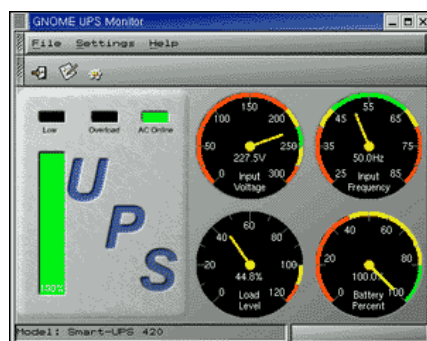
Na záložke **Panel** definujeme, aké parametre sa budú zobrazovať v ľavej časti okna – obr.č.12:



Po kliknutí na tlačidlo *OK* môžeme zobrazovať uvedené parametre v peknom grafickom prevedení – obr.č.13:



Pre prostredie Gnome je veľmi známy klient **gupsc**. Je vzhľad je na obrázku č.14:



Tááák, server máme po energetickej stránke zaistený, takže môžeme spokojnejšie spávať. A že sa nám to podarilo, o tom môžeme poslať našim priateľom mejl.

Čože, že my ešte nemáme funkčný poštový systém?

Takže, rýchlo na to! Ale až nabudúce!

Linux prakticky ako server / 19.časť

„Jééééde jééééde poštový panáááček, jéééde jéééde poštový páádán...“

Voľakedy dávno sa touto znelkou za pomoci trúbky (ktorej sa hovorí aj poľnica) oznamovalo doručenie pošty do obce či mesta. A práve preto sa táto trúbka stala symbolom poštových služieb. No nie o poštovej trúbke, ale o pošte (zvlášť o tej elektronickej) sa dnes budeme rozprávať.

Pošta

Tí, čo vytvorili princíp elektronickej pošty boli veľmi múdri chlapíci. Aby sa nemuseli namáhať, odpozorovali klasickú poštu a tú len previedli do elektronickej podoby.

Aby sme tej elektronickej podobe pošty dobre porozumeli, vráťme sa na chvíľu ku klasickej pošte.

Poštová poviedka

Keď chcela pani Leonka z Banskej Bystrice oznámiť svojej kamarátke Margitke zo Zvolena určitú udalosť, sadla si k stolu a napísala list. Ten list vložila do obálky, na ktorú napísala meno a adresu – pre jednoduchosť iba stručne „Margitka zo Zvolena“. Privolala svoju vnučku Mirku, aby list zanesla na poštu. Mírka išla do mesta a tam odovzdala list poštovému úradu v BB.

Poštový úrad „BB“ prevzal list, prečítal adresu a Leonkin list vložil do vreca, smerujúceho do Zvolena. Potom úrad odoslal toto vrece poštovému úradu „ZV“ vo Zvolene. Poštový úrad „ZV“ vybral z vreca list a odovzdal ho poštovému doručovateľovi Karolovi. Ten prezrel obálku, či nie je porušená, skontroloval, či neobsahuje niečo nevhodné, sadol na bicykel, prešiel polovicu Zvolena a list vhodil do poštovej schránky s menom „Margitka“. Margitkina vnučka Elenka po návrate zo školy schránku otvorila, list vybrala a doniesla staršej Margitke. To bolo ale radosti, že si na nich Leonka spomenula.

A teraz skúsme tento príbeh prepísať do modernej podoby.

Elektronická pošta

Keď chce pani Leonka z Banskej Bystrice oznámiť svojej kamarátke Margitke zo Zvolena určitú udalosť, sadne si za svoj domáci počítač a v príslušnom programe, ktorý sa označuje **MUA** – *Mail User Agent* - napíše správu. Napíše adresu v tvare margitka@zvolen.sk, predmet správy a nakoniec samotný obsah. Jej poštový program **MUA** správu prevezme a cez zriadené spojenie poštu odovzdá poštovému serverovému programu „BB“, ktorý sa označuje **MTA** – *Mail Transfer Agent*. Poštový program **MTA** „BB“ si preštuduje adresu príjemcu, hlavne jej časť napravo od zavináča („zvolen.sk“) a prepošle ho inému poštovému programu **MTA** „ZV“, ktorý je príjemcom správ pre doménu @zvolen.sk. Poštový program **MTA** „ZV“ preberie zásielku, preštuduje si adresu, ale tú časť naľavo od zavináča („margitka“) a buď ju sám doručí alebo ju odovzdá inému programu, ktorý sa označuje **MDA** – *Mail Delivery Agent*. Tento **MDA** prevezme správu, vykoná rôzne filtrácie a previerky a potom vloží správu do priečinka s názvom „margitka“. Tam správa čaká, až si ju vyzdvihne program **MUA** na Margitkinom domácom počítači, v ktorom si správu Margitka prečíta.

Keď sa na to pozrieme podrobnejšie, vidíme jasnú analógiu so svetom klasickej pošty.

Poštové programy

Pozrime sa teraz na jednotlivé poštové programy podrobnejšie:

MUA – Mail User Agent je program, v ktorom sa správy tvoria a čítajú. Vôbec nezáleží, na akej platforme tento program beží.

Medzi najznámejšie programy typu MUA v prostredí MS Windows patria tieto programy:

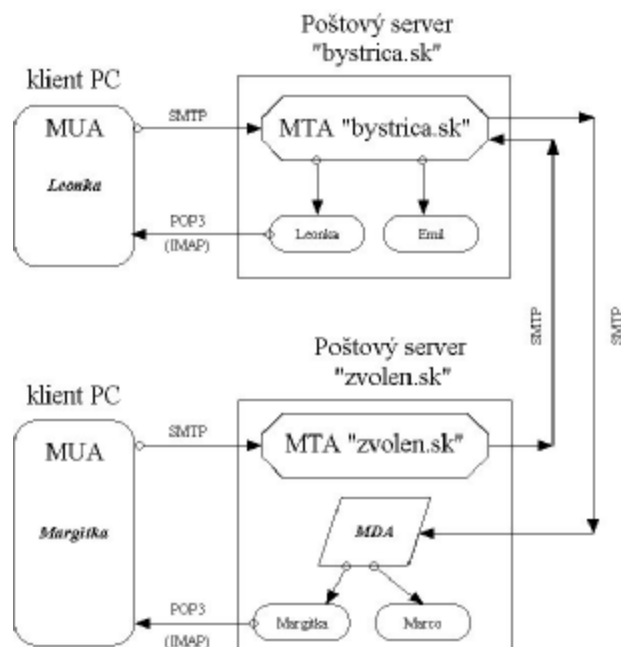
- Ø MS Outlook
- Ø MS Outlook Express
- Ø Netscape Navigator
- Ø Opera (mailová časť)
- Ø ThunderBird
- Ø a iné

V prostredí Linuxu sú to tieto programy:

- Ø Opera for Linux (mailová časť)

- Ø ThunderBird
- Ø Kmail
- Ø Mutt
- Ø Pine
- Ø Evolution
- Ø a kopec iných dobrých programov

Teraz sa pozrime na obrázok č.1:



MUA , okrem toho, že umožňuje písanie a čítanie správ vo viac – či menej komfortnom prostredí, má dve dôležité úlohy:

Prvou úlohou MUA je prevziať od odosielateľa správy text a adresu príjemcu, pridať základnú hlavičku a celé to pomocou normalizovaného protokolu *SMTP (Simple Mail Transport Protocol)* odovzdať programu MTA na odoslanie ďalej.

Druhou úlohou MUA je z používateľovej elektronickej poštovej schránky za pomoci protokolu *POP3 (Post Office Protocol ver.3)* alebo *IMAP (Internet Message Access Protocol)* vyzdvihnúť poštové správy a zobraziť ich na obrazovke počítača.

Ak sa vrátíme k poštovej poviedke, tak MUA je vlastne Mirka alebo Elenka v elektronickej podobe.

MUA spravidla beží na klientskom počítači používateľa. Ako sme si povedali, je jedno, pod akou platformou a aký MUA program používame, dôležité je, aby dodržiaval dohodnuté protokoly SMTP, POP3 alebo IMAP. Vedzme ale, že najčastejšie to (ešte stále) býva platforma MS Windows.

MUA je spojený s poštovým serverom cez zriadené pripojenie, napríklad cez vytáčanú linku pomocou modemu, cez pevnú linku, mikrovlnu alebo sieť LAN. MUA môže byť od servera vzdialený len niekoľko metrov ale aj tisíce kilometrov. Nevadí – dôležité sú protokoly!

MTA – Mail Transfer Agent

Na rozdiel od MUA, MTA beží na niektorom zo serverov - u nás samozrejme na Linuxe. Na tomto linuxovom serveri sú zároveň vytvorené poštové schránky jednotlivých používateľov. Poštová schránka je v podstate súbor (alebo adresár so súbormi), kde sa jednotlivé správy ukladajú po doručení.

MTA je program, ktorý je zodpovedný za prepravu elektronickej pošty v rámci jeho pôsobnosti. Pôsobnosť MTA je daná doménou, ktorú spravuje, napr. *bystrica.sk*.

MTA má akoby dve časti – vysielaciu a prijímaciu.

Vysielacia časť preberá poštu od MUA a prešľahuje si adresu domény (napravo od zavináča) príjemcu pošty. Ak sa pravá časť adresy zhoduje s vlastnou doménou (*bystrica.sk*), a ak sa meno príjemcu zhoduje s názvom

niektorej poštovej schránky (napr. *emil*), tak správu presunie do tejto poštovej schránky svojho (lokálneho) používateľa.

Ak sa príjemca nachádza mimo jeho doménu, vysielacia časť MTA pomocou služby DNS zistí, kto je zodpovedný za príjem pošty pre doménu príjemcu (napr. *zvolen.sk*). Keď toto zistí, pomocou protokolu SMTP odošle poštu danému MTA (pre doménu *zvolen.sk*).

Príjímacia časť MTA preberá poštu od iných MTA. Preštuduje si adresu príjemcu, hlavne jej časť pred zavináčom. Ak nájde okolo seba poštovú schránku s rovnakým menom (v našom prípade „margitka“), poštu uloží do tejto schránky. Ak nenájde schránku daného mena, podľa dopredu stanovených pravidiel vykoná príslušnú operáciu s poštou (zahodí ju alebo ju vráti alebo prepošle inému používateľovi).

V zmysle poštovej poviedky je MTA v podstate poštový úrad „BB“ a „ZV“ v elektronickej podobe.

Medzi najznámejšie MTA patria programy **sendmail**, **qmail** a **postfix**.

Sendmail je veľmi robustný program, ktorý do Linuxu prišiel zo sveta Unixu. Jeho konfigurácia je pomerne zložitá a o *sendmail* sa traduje, že sa konfiguruje iba raz. Nie preto, že by to stačilo, ale že po konfigurácii *sendmailu* skončil správca poštového servera v blázinci.

Qmail a *postfix* sú podstatne jednoduchšie na konfiguráciu a pritom dosahujú zrovnateľných, ak nie aj lepších kvalít ako *sendmail*. (My sa budeme v našom seriáli venovať postfixu).

Jednotlivé MTA komunikujú medzi sebou zásadne protokolom SMTP (alebo novším ESMTP).

MDA – Mail Delivery Agent

Môže sa stať, že na ceste medzi MTA a poštovou schránkou príjemcu na príjímacej strane leží ešte MDA (pozri poštový server „zvolen.sk“ na obrázku č.1) . Jeho úlohou je vykonať s prijatou poštou určité operácie, napríklad skontrolovať na prítomnosť vírusov, odhaliť *spam* (*spam* = nevyžiadaná, obťažujúca pošta) alebo aplikovať celú radu rôznych filtrov, presmerovať do inej schránky, preposlať na mobilný telefón či vytvoriť automatickú odpoveď o doručení. Po vykonaní svojej úlohy MDA doručí správu priamo do poštovej schránky príjemcu.

Asi najznámejší a najpoužívanejší MDA je program *procmail*.

Prítomnosť MDA v systéme však nemusí byť nutnosťou a závisí od ostatných okolností, či chceme ešte poštu spracovávať alebo nie. Napríklad v poštovom serveri „bystrica.sk“ na obr.č.1 sa o doručovanie pošty do lokálnych poštových schránok postará priamo MTA.

Z analógie s poviedkou už vieme, že MDA je taký elektronický poštový doručovateľ Karol a preskúmanie obálky sú aplikované filtre a pravidlá.

Príjem pošty

Povedali sme si, že po doručení (delivery = doručenie) do poštovej schránky používateľa je potrebné túto poštu ešte zo serveru vyzdvihnúť a preniesť do MUA k prečítaniu.

Na vyzdvihovanie pošty slúžia dva protokoly – POP3 a IMAP.

POP3

POP3 je tretia verzia *Post Office Protocol*-u. Je to jednosmerný protokol a slúži iba k jedinému účelu – vyzdvihovaniu pošty zo schránky na serveri a prenos do lokálneho počítača používateľa.

MUA sa pomocou POP3 protokolu skontaktuje s poštovým serverom, kde sú uložené schránky jednotlivých používateľov. Po overení menom a heslom sa pošta v schránke vyzdvihne.

POP3 protokol môže byť nastavený tak, aby po prenose správ bola poštová schránka vyprázdnená. Všetky správy sa teraz nachádzajú v lokálnom počítači.

POP3 protokol však môže byť nastavený aj tak, aby poštovú schránku po prenose nevyprázdňoval, ale správy zanechal po určitú dobu alebo nastalo v schránke. Ponechané správy dostanú príznak, že už boli raz prenesené a tak sa nestane, že by sa pri novom prihlásení z toho istého MUA stiahli ešte raz.

Bohužiaľ, táto voľba veľmi zaťažuje poštové servery, lebo pri veľkom počte spravovaných poštových účtov dochádza veľmi rýchlo k zaplneniu diskov servera (a preto ich automaticky po nejakej dobe maže server sám). Preto treba použitie tejto voľby zvážiť.

Na používanie POP3 protokolu je nutné, aby sa na poštovom serveri nachádzali schránky v príslušnom formáte. Tomuto formátu sa hovorí **mailbox**. Je to v podstate jeden súbor, ktorý sa nachádza spravidla v adresári

/var/spool/mail/ a jeho meno je identické s menom používateľa. Pre Margitku to teda bude súbor /var/spool/mail/margitka.

Všetky správy sa ukladajú do tohto jediného súboru a to vrátane príloh. Takže ak Leonka pošle Margitke fotky svojich všetkých vnúcat, jedinou správou môže tento súbor narásť o desiatky megabajtov.

K *mailboxu* je možné pristupovať nielen z klientského počítača, kde máme daným spôsobom nastavený MUA, ale aj z iných počítačov, napríklad v kancelárii, kde si tam nainštalovaný MUA nastavíme. Musíme si ale uvedomiť, že si môžeme prečítať iba novo došlé správy, a ak tieto stiahneme do počítača v kancelárii, doma ich už neuvidíme. Takisto sa nemôžeme vracieť k starším správam, lebo jedny sú stiahnuté do kancelárskeho a druhé do domáceho počítača (len tak na okraj, aj toto sa dá obísť. Ale o tom inokedy...).

Aby sa tieto nedostatky aspoň trochu eliminovali, bola vytvorená nová služba **WebMail**. Tá umožňuje, aby sme správy na poštovom serveri mohli čítať kdekoľvek, trebárs aj pomocou Internetu. Stále tu však hrozí riziko, že ak raz tieto správy stiahneme pomocou POP3, cez WebMail uvidíme iba novo prichádzajúce správy.

IMAP

IMAP (dnes vo verzii 4) je dokonalejšou obdobou protokolu POP3. Základný rozdiel je ten, že všetky správy zostávajú v schránke používateľa na serveri – teda sa do klientského počítača neprenášajú! Používateľ môže tieto správy triediť, prezerať alebo mazať bez toho, aby k sebe prenášal megabajty dát. Je to výhodné zvlášť pre spojenie cez pomalé linky, keď si môžeme prečítať obsah správy bez toho, aby sme k sebe ťahali aj niekoľkomegovú prílohu.

Najväčšou prednosťou IMAP protokolu je, že používateľ môže pristupovať k svojej pošte z rôznych miest na svete a VŽDY bude mať k dispozícii rovnaké správy, nové aj staré, ktoré už čítal.

Aby sme mohli používať IMAP, musíme poštovému serveru povedať, aby schránky vytváral vo formáte pre tento protokol. Tomuto formátu sa hovorí **maildir**, a správy sa ukladajú ako jednotlivé súbory do jednotlivých adresárov. Čo jedna správa, to jeden súbor a jeden adresár.

Okrem tejto prednosti však IMAP trpí veľkým problémom. Keďže sa správy zanechávajú na serveri a nie je možné ich mazanie, poštový server musí mať dostatočnú diskovú kapacitu.

A je tu ešte jeden háčik – nie všetky klientské MUA vedia s *maildirom* spolupracovať!

Preto si pri tvorbe poštového servera treba všetko dobre zvážiť a správne sa rozhodnúť, aký protokol použijeme. Lebo potom už nie je cesty späť (alebo to aspoň nie je vôbec jednoduché).

My si v našom seriáli ukážeme všetky tri možnosti – POP3, IMAP aj WebMail.

Bezpečnosť

Celý proces prenosu pošty je v podstate nezabezpečený. Pošta sa prenáša ako prostý (plain = čistý) text.

Potencionálny útočník môže správu zachytiť a prečítať.

Preto nastala potreba, aby sa správy po ceste šifrovali.

Prenos POP3 a IMAP je možné vykonať po šifrovanom kanále (POP3 over SSL), alebo použijeme SSH.

Takisto SMTP protokol môžeme vykonať cez SSL/TSL.

POP3 autentizáciu môžeme nahradiť autentizáciou APOP, RPOP, KPOP alebo inou.

Všetky tieto metódy vyžadujú podporu ako na strane klienta tak aj servera.

Univerzálnou metódou ochrany správ a (dát všeobecne) je šifrovanie asymetrickou šifrou. Najpoužívanejším riešením tohto druhu je *PGP*.

My si niektoré z tu spomínaných metód postupne ukážeme.

Variety elektronickej pošty

Tak ako samotné ľudské bytie, tak aj poštové spojenie a elektronickej komunikácia nie je jednoznačná. Existuje niekoľko variantov elektronickej pošty. Tieto varianty vychádzajú z potrieb ľudí, firiem a organizácií, komunikujúcich pomocou elektronickej pošty.

Vyššie opísaný princíp elektronickej pošty je akýmsi základom jednotlivých variantov.

Nabudúce si vysvetlíme najpoužívanejšie varianty a potom si to všetko vyskúšame na praktických príkladoch.

Linux prakticky ako server / 20.časť

V minulej časti sme si čo-to povedali o základoch a princípe elektronickej pošty. Porovnali sme ju s klasickou poštou, ale hlavne sme si vysvetlili, z akých súčastí sa pošta skladá. Na záver sme si povedali, že existuje niekoľko variantov elektronickej pošty. A dnes si tieto varianty pekne rozoberieme.

Varianty elektronickej pošty

Možno niekto namietne, že sa až príliš dopodrobna zaoberáme poštou. Na to mám hneď niekoľko argumentov: Po prvé, pošta (v akejkoľvek podobe, či klasickej alebo elektronickej) bola, je a bude veľmi dôležitým komunikačným prvkom v spoločnosti.

Po druhé, elektronická pošta bola to, čo dalo podnet na vznik Internetu (nie, nie moji milí, neboli to web stránky, ale elektronická pošta – to boli počiatky Internetu).

Po tretie, pošta je pomerne komplikovaná vec, a ak chceme niečo fundovane navrhnúť, zriadiť a prevádzkovať, musíme tomu dobre rozumieť. A tu uvedené varianty sú skúsenosti z praxe.

Keďže sa pošta skladá z niekoľkých vzájomne prepojených súčastí, môže nastať dostatok variácií. Ale na základe praxe môžeme rozdeliť poшту do troch základných skupín – variantov:

- Ø **koncový klient**
- Ø **systém s nepriamym doručovaním pošty**
- Ø **systém s priamym doručovaním pošty**

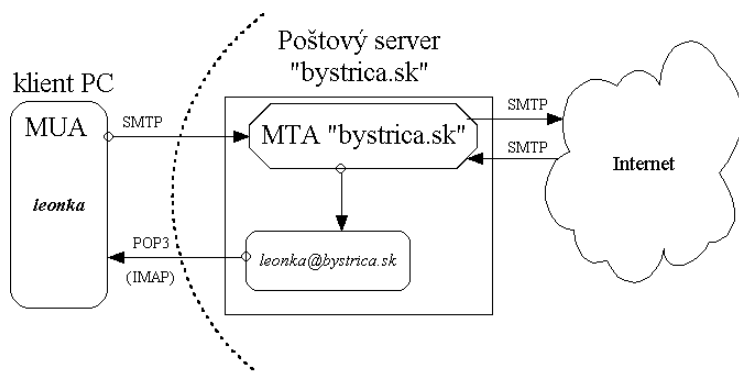
Musíme mať na pamäti, že v každom variante môže existovať niekoľko drobných odchýlok, na ktoré sa budem snažiť upozorniť, ale podstata jednotlivých variantov zostane zachovaná.

Podme sa teraz bližšie pozrieť na jednotlivé varianty:

Variant „koncový klient“

Variant „samostatný klient“ je asi najrozšírenejší zo všetkých tu spomenutých variantov.

Na obrázku č.1 je zobrazená schéma takéhoto typu pripojenia:



Bežný používateľ poštových služieb si objedná pripojenie (najčastejšie z domu) k danému poskytovateľovi služby elektronickej pošty.

Nech je pre ilustráciu poskytovateľom služby firma XYZ, ktorá spravuje poštový server v doméne *bystrica.sk*.

Tento server má aj svoje meno a spravidla to býva napr. *mail.bystrica.sk* alebo *pop3.bystrica.sk* a podobne.

Keď klient požiada o poštovú službu, poskytovateľ vytvorí na serveri používateľovi účet (buď skutočný alebo alias – ešte si vysvetlíme) a poštovú schránku s požadovaným menom. Nech pre náš prípad je používateľom pani Leona O. a jej účet je pomenovaný *leonka*. Je zrejmé, že emajlová adresa bude *leonka@bystrica.sk*.

Poskytovateľ poskytne klientovi tieto údaje:

- Ø prihlasovacie meno (spravidla identické s názvom účtu), napr. *leonka*
- Ø prihlasovacie heslo, napr. *7CeV3ro*
- Ø meno servera na odosielanie pošty protokolom SMTP, napr. *smtp.bystrica.sk* (alebo *mail.bystrica.sk*)
- Ø meno servera na príjem pošty protokolom POP3 alebo IMAP, napr. *pop3.bystrica.sk* (ale aj totožné so SMTP serverom, teda *mail.bystrica.sk*)
- Ø typ protokolu pre príjem pošty, napr. POP3 (alebo aj IMAP)
- Ø adresy portov jednotlivých služieb, ak sú odlišné od defaultne nastavených.

Poznámka:

Defaultné čísla portov pre jednotlivé služby sú:

- Ø SMTP na porte 25
- Ø POP3 na porte 110
- Ø IMAP na porte 143

Vyššie spomenuté údaje musíme nastaviť na strane klienta v programe MUA.

Po zadaní sa môžeme komunikačným kanálom spojiť s našim poskytovateľom poštovej služby a odosielať alebo prijímať poštu.

Teraz nezáleží, o aký komunikačný kanál sa jedná. V prípade domácich používateľov to najčastejšie býva vytáčané spojenie pomocou modemu a telefónnej linky, ale dnes už nebýva zvláštnosťou pripojenie pomocou mikrovlnného spoja Wi-Fi alebo káblovým modemom, ADSL a podobne.

Je zrejmé, že vzdialenosť používateľa od poskytovateľa nie je podstatná a závisí iba od použitého komunikačného kanála.

Vyššie spomínaný príklad a obrázok č.1 je typickým prípadom poštového spojenia pre domáce použitie. Jeho podstatou je, že u poskytovateľa poštových služieb musíme mať „napevno“ definovaný účet s adresou leonka@bystrica.sk alebo margitka@zvolen.sk. Čo jedno meno, to jeden účet!

Viem, že tento variant je viac-menej z pohľadu klienta, nie servera, ale nedá mi ho tu nespomenúť. A to z dôvodu svojej podstaty – každý náš používateľ je v podstate koncový klient!

Aj keď sa tento variant najčastejšie používa v domácom prostredí, je vhodný aj pre menšie – rodinné firmy s veľmi malým počtom používateľov – klientov. U poskytovateľa pripojenia objednáme určitý počet poštových schránok a každému nášmu používateľovi v sieti nastavíme jeho MUA tak, ako sme si to popísali vyššie. Bohužiaľ, v takomto prípade sme odkázaní na to, čo nám poskytne provider. Myslím tým dodatkové služby, ako je antivírusová a antispamová ochrana, preposielanie, kopírovanie mejlov a iné drobnosti, ktoré sú dnes už štandardom.

Taktiež so zvyšujúcim počtom poštových schránok na strane poskytovateľa narastá cena za túto službu a zrazu zistíme, že sa nám viac vyplatí mať vlastný poštový server. A to je prípad ostatných variantov.

Variant „systém s nepriamym doručovaním pošty“

Predstavme si, že sme správcami IT v určitej firme FIRMA, ktorá je pripojená do Internetu (zase nezáleží ako) a máme zabezpečiť poštové spojenie.

Keďže počet našich používateľov je pomerne veľký a ešte k tomu sa často menia, jedni prichádzajú druhí odchádzajú, variant samostatného klienta s x-počtom samostatných poštových schránok neprípadá v úvahu. Čo teraz?

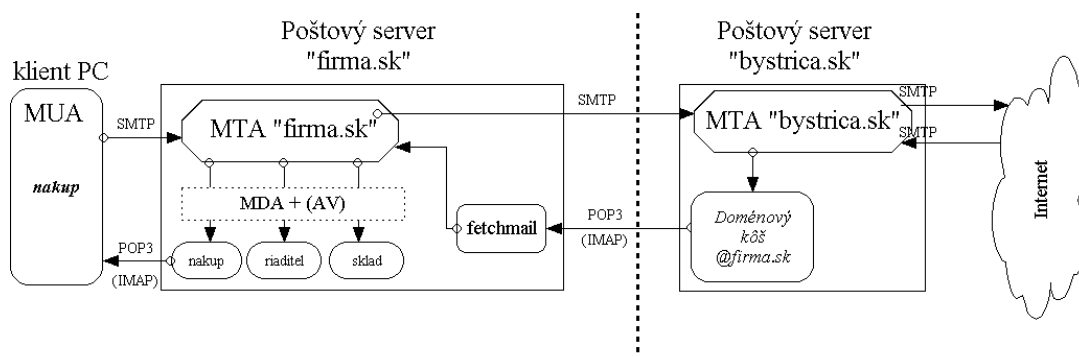
Pre takýto prípad existuje lepšia metóda – **doménový kôš**.

Doménový kôš je jedna poštová schránka, ktorá sa nachádza u poskytovateľa poštových služieb a má jedinečnú vlastnosť. Prijíma poštu, kde dôležitým parametrom nie je meno príjemcu, ale doména príjemcu (doména je označenie skupiny počítačov spoločným menom). Doménový kôš je v podstate jeden účet pre neobmedzený počet používateľov.

Aby sme mohli používať doménový kôš, musíme mať vlastný poštový úrad - server!

Náš poštový server bude akýsi medzičlánok medzi providerom (kde je uložený doménový kôš) a našimi používateľmi – príjemcami pošty.

Pozrime sa na obrázok č.2:



Príklad použitia doménového koša

Necháme si pre našu firmu zaregistrovať doménu s názvom **firma.sk**. Aby sme doménu *firma.sk* získali, musíme najprv zistiť, či už niekto doménu s týmto menom nevlastní a ak nie, tak požiadame o jej registráciu. Zistenie aj registráciu domény všeobecne môžeme vykonať na www.sk-nic.sk alebo cestou tam spomínaných registrátorov. Veľmi často sa stáva, že náš poskytovateľ pripojenia je zároveň aj registrátorom domén.

U poskytovateľa poštových služieb a Internetu si necháme zriadiť doménový koš pre našu doménu *firma.sk*. (Teraz vôbec nezáleží na tom, že náš poskytovateľ (provider) má doménu *bystrica.sk*!)

Odteraz, ak niekto pošle správu s doménovou adresou @firma.sk, táto pošta sa objaví v našom doménovom koši. Ak máme teda vo firme používateľov *nakup*, *riaditel* a *sklad*, ich emailové adresy budú nakup@firma.sk, riaditel@firma.sk a sklad@firma.sk.

Ak nám časom pribudne používateľ *predaj*, jeho mejlová adresa bude predaj@firma.sk. Poskytovateľ a vôbec nezaujíma, koľko máme používateľov, u neho sa nič nemení!

Ako to funguje?

Ak niekto pošle správu s adresou nakup@firma.sk, táto sa najprv doručí až na poštový server *bystrica.sk*. Tento server podľa svojich nastavení zistí, že uvedená správa patrí pre firmu FIRMA a tak túto správu uloží do doménového koša @firma.sk. Tam bude pošta čakať, pokiaľ ju nejakým spôsobom nevyberieme.

Z pohľadu servera *bystrica.sk* sa doménový koš javí ako špecifická poštová schránka a tak doručenie pošty do koša jeho práca končí.

Zároveň sa môže ale stať, že niekto pošle poštu s adresou direktor@firma.sk a my takého používateľa nemáme! Čo sa stane?

Nič!

Táto správa sa doručí do doménového koša tak ako tie ostatné. **Pre doménový koš nie je rozhodujúce meno príjemcu, ale názov jeho domény.**

Takto si firmy a organizácie môžu vytvárať ľubovoľný počet mejlových adries.

Náš poštový server sa bude skladať z programu MTA, v našom prípade *postfix* a programu *fetchmail*. Na poštovom serveri vytvoríme účty pre našich používateľov.

Povedali sme si, že doménový koš je zvláštny typ poštovej schránky a tak do neho doručená pošta v ňom skončí svoju púť.

Preto našou úlohou je, aby sme zabezpečili vyberanie doménového koša v pravidelných intervaloch, teda presunutie uloženej pošty na náš poštový server k ďalšiemu spracovaniu.

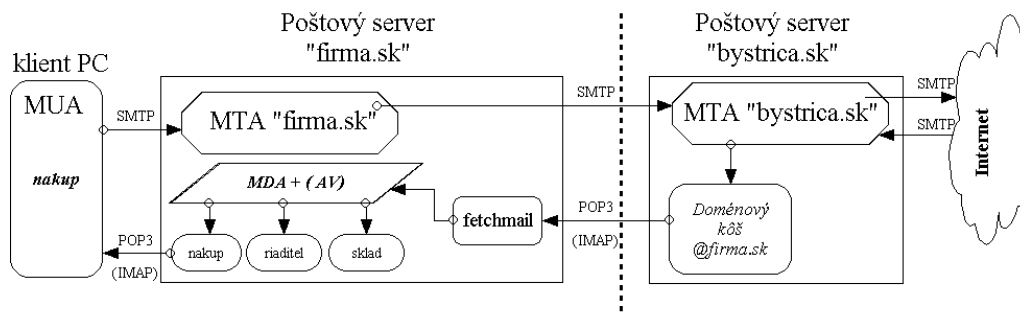
A práve na to použijeme program **fetchmail**.

To je program, ktorý dokáže vybrať poštu z doménového koša (alebo iného poštového priečinku) pomocou protokolu POP3 alebo IMAP.

Takto vybranú poštu odovzdá programu MTA *postfix-u*. Ten prijatú poštu roztriadi do jednotlivých priečinkov podľa mien používateľov a prípadne vykoná antivírusovú kontrolu (AV).

V priečinkoch používateľov bude pošta čakať na vyzdvihnutie klientským programom MUA.

Program MTA (*postfix*) nemá veľmi široké možnosti manipulácie s poštou. Preto v niektorých prípadoch môžeme doručiť poštu do priečinkov používateľov pomocou MDA, napríklad *procmail* a podobne. Obdobu tohto variantu ukazuje obrázok č.3:



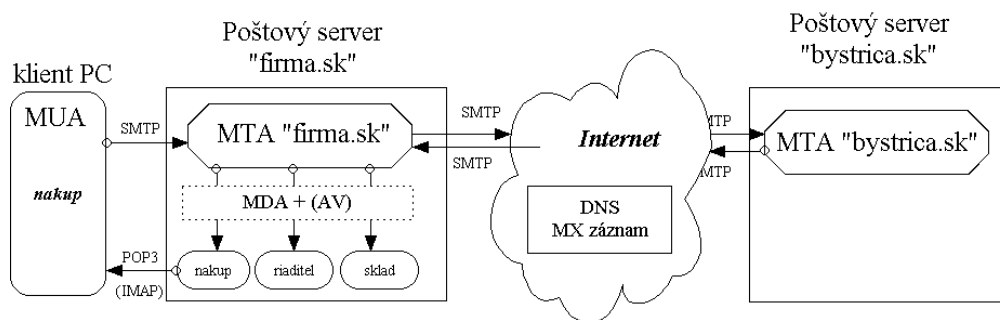
V tomto prípade *fetchmail* stiahne poštu z doménového koša, ale neodovzdá ju MTA (*postfixu*), ale predá ju MDA (*procmailu*) na ďalšie spracovanie. A ten – po vykonaní príslušných filtrácií – potom doručí poštu do schránok používateľov.

Klientské počítače našich používateľov nastavíme podobne, ako to je popísané vo variante *koncového klienta*.

Tento variant je vhodný pre malé a stredné firmy. Pre niektoré situácie však ani doménový kôš nie je vhodným riešením. Preto existuje iný variant.

Variant „systém s priamym doručovaním pošty“

Podstata tohto variantu je na obrázku č.4:



Tak ako v predchádzajúcom prípade, aj tu si musíme zabezpečiť vlastnú doménu *firma.sk*.

Ak už máme doménu *firma.sk* zaregistrovanú, od správcu domény *.sk* (alebo poskytovateľa pripojenia do Internetu) si vyžiadame verejnú IP adresu. (Len na okraj pripomínam, že IP adresy sa delia na verejné – to sú tie, čo sú v Internete vidieť a na neverejné – to sú tie, čo sa nesmú v Internete objaviť. Neverejné adresy sa používajú v lokálnych sieťach – o tom inokedy).

Zároveň správcu domény *.sk* požiadame, aby na príslušnom serveri DNS vytvoril MX záznam pre našu doménu. *MX záznam* je zápis o tom, kto je príjemcom pošty pre určitú doménu. V našom prípade budeme chcieť, aby MX záznam poukazoval na to, že pre doménu *@firma.sk* je príjemcom náš poštový server s priradenou (verejnou) IP adresou.

Týmto zabezpečíme, že sa akákoľvek pošta s menom *@firma.sk* doručí priamo na náš poštový server.

Odtiaľ sme dosiahnuteľní vo svete internetovej pošty.

Všimnime si, že pri tomto variante už nie sme závislí od poštového úradu *bystrica.sk*, ale sme rovnoprávne postavení v internetovej hierarchii.

Na vybranom počítači (serveri) nainštalujeme poštový program MTA (*postfix*) a vytvoríme účty pre našich používateľov. Môžeme použiť aj programy pre doručovanie (MDA) a pre antivírusovú kontrolu (AV), ktoré nastavíme tak, aby spolupracovali s programom *postfix*.

Nakoniec nastavíme klientské počítače našich používateľov a sme za vodou.

Naše MTA sa postará o to, aby odchádzajúca pošta bola okamžite odoslaná príjemcovi a naopak, ak sa z Internetu doručí pošta pre nášho používateľa, bude mu okamžite doručená do poštovej schránky. Tam si ju klient v nastavených intervaloch vyberie.

Dôležitou vecou je, že náš poštový server je vo svete Internetu verejne známy, teda jeho meno a IP adresa je dostupná pre všetkých účastníkov Internetu!

Z vyššie uvedených charakteristík je zrejmé, že tento variant je vhodný pre veľké firmy s veľkým počtom používateľov.

Tak, dosť teórie, nabudúce začneme stavať náš vlastný poštový server.

Miroslav Oravec