UNIX

Radek Hnilica

Sestavil
Radek Hnilica

UNIX

Radek Hnilica

Sestavil Radek Hnilica

Working Vydání
Vydáno pubdate:

Overview

Tento dokument vznikal postupně jako poznámkový sešit. Psal jsem si sem na co jsem došel. Krátké ukázky použití programů, jejich konfigurace a odkazy na web kde jsou relevantní informace k danému programu. Některé části jsou velmi sporé a obsahují jen odkazy či seznam programů/možností, jiné jsou rozpracovány hlouběji. Jak šel čas, tak jsem několikrát, za odlišných okolností přišel o část práce. Některé části se proto ztratily, či jsou v jiném formátu či kódování v adresáři `input`. Dokument je stále udržován, podle možností a času. Ale někdy se toho času nedostává a pak se dlouho nic nemění. Mnohokrát jsem si říkal, že musím některá témata přepracovat, či dokonce vůbec napsat v případě že jsem do nich napsal jen odkazy na web.

Doufám, že tento dokument bude někomu ku prospěchu.

Náplní tohoto dokumentu je hlavně Linux, v distribuci Debian. Některé informace jsou ale obecnějšícho charakteru, a tak se dají aplikovat i na jiné distribuce Linuxu, či dokonce na jiné varianty UNIXu obecně. Některé části se dokonce Linuxu či UNIXu nemusí týkat vůbec, či jen vzdáleně. Název dokumentu UNIX tak nemusí být zcela přesný, ale zatím mi postačuje a nikdo k němu neměl připomínky.

Tento dokument je primárně publikován ve dvou formách, jako statické webové stránky, a ve formátu vhodném pro tisk. Jako formát vhodný pro tisk rozumím Postscript nebo PDF. Ve verzi pro tisk dokument čítá 619 stran. Toto číslo je dané valikostí tiskového zrcadla, jakožto i velikostí fontů a dalších nastavení. Toto číslo vyčítám z mnou generovaného postscriptového souboru. Pokud by byl dokument formátován pro tisk za jiných podmínek, počet stran se může lišit.

Příšerně žluťoučký kůň úpěl ďábelské ódy.

Hleď, toť přízračný kůň v mátožné póze šíleně úpí.

Počet stran: 619

Přehled revizí

Revize 2 2009-12-16 03:49 CET Revidoval: rfh
Po delší době jsem dokument opět zavedl do SCM systému. Začínám do něj zahrnovat části ze starších "ztracených" verzí.
Revize Revision: 1.7 Date: 2009-03-07 03:52:40 Revidoval: rfh
Aktuální pracovní verze.
Revize 0 2007-07-12 Revidoval: rfh
Publikováno

Věnování

FIXME:napsat Dedication.

Obsah

Seznam tabulek

Tiráž

FIXME:napsat Colophon.

Předmluva

* ***FIXME:****Zpracovat text této předmluvy do kapitoly.*
* ***FIXME:****Sehnat někoho kdo mi napíše předmluvu.*

Tak jak jde čas, pocítil jsem potřebu systematizovat alespoň částečně své poznámky, které jsem si nikdy pořádně nevedl. Výsledkem je tento nesourodý text. Snažil jsem se, a snažím se, zapisovat si postupy které jsem použil, a informace které jsem získal. Značná část textu je velmi prostá a je mi jen vodítkem a připomínkou jak jsem co dělal. Někde jsem se ale snažil a trochu se i rozepsal v celých větách.

Tento dokument je k dispozici v několika různých formátech. Jako vícestránkový HTML dokument (index.html), postsriptový (unix.ps) či PDF (unix.pdf) soubor formátovaný na velikost papíru A4. HTML verzi je možno získat také zabalenou () pro offline čtení.

* ***FIXME:****Přidat do výstupního adresáře balíček všech html stránek.*
* ***FIXME:****Přidat do výstupního adresáře také Postsript a PDF.*

Poznámky autora jsou jen v autorské verzi dokumentu. Nejsou určeny k publikaci, ale popisují věci související s vytvářením dokuemntu.

Tato kniha vznikala postupně v průběhu mnoha let, začal jsem na ní pracovat někdy v roce 2001, kdy jsem vzal všechny své poznámky vytvořené v předchozích letech a převedl je do formátu DocBook/XML (tehdy ještě SGML).

Tento dokument je souhrnem více či méně souvislých poznámek a textů hlavně u LINUXu ale i jiných systémech. Byl vytvořen v průběhu mnoha let a ve své podstatě je to poznámkový sešit, jenž jsem psal a píši hlavně pro svou vlastní potřebu.

V průběhu práce na tomto dokumentu došlo k několika „katastrofám“. Byly to přechody mezi verzemi Linuxu a změny v programech které ovlivnily způsob zpracování DocBook dokumentace. A také to byla ztráta zdrojových souborů zapříčiněná selháním disku.

* ***To be done:****Následují šablony sekcí.*

1. Foreword

* ***To be done:***

2. Audience

* ***To be done:***

3. Prerequisities

* ***To be done:***

4. Organizace částí knihy

* ***To be done:***

Tato kniha je rozdělena do částí, které se snaží držet jednoho tématu. Tyto části jsou:

- I – „*Historie operačních systémů převážně UNIXového typu*“ v *UNIX*
- II – „*Správa UNIXu*“ v *UNIX*
- Sítě a vše okolo — konfigurace sledování
- Síťové služvy — mail, …
- Nástroje
- X Window
- Programování
- Programování v jazyce C
-
-

Na začátku každé části, kapitoly, sekce či podsekce uvádím seznam odkazů ze kterých jsem čerpal, nebo které s popisovanou tématikou souvisí a jsou vhodné k dalšímu studiu. Vzhledem ke sporému textu jsou tyto odkazy někdy jediné co sekci tvoří.

Kapitola 1. Úvod

FIXME: napsat úvod.

I. Historie operačních systémů převážně UNIXového typu

Tato část knihy je kompletně prázdná. Mám ji zde jen jako připomínku toho, co bych měl doplnit.

Odkazy:

- Operating system (http://en.wikipedia.org/wiki/Operating_system)

Pořadí v kterém jsou jednotlivé operační systémy je náhodné. Prostě jak jsem si na ně vzpoměl, tak jsem je napsal.

Kapitola 2. Historie

* *Kapitola je ve velmi rozpracovaném stavu. Spíše je připomínkou toho ce je ještě třeba dospat.*

Nezpracované odkazy:

- History of UNIX (http://www.dei.isep.ipp.pt/docs/unix-Contents.html)
- The UNIX Industry: A Brief History (http://snap.nlc.dcccd.edu/learn/drkelly/brf-hist.htm)
- The Creation of the UNIX* Operating System (http://www.bell-labs.com/history/unix/)
- The First Unix Port (https://db.usenix.org/publications/library/proceedings/usenix98/invited_talks/miller.ps) by Richard Miller
- Version 6 Unix (http://en.wikipedia.org/wiki/Version_6_Unix) na Wikipedii

UNIX vznikl nekdy v roce 1969 a od té doby se nepřetržitě vyvýjí.

Odkazy:

- The Unix Heritage Society (http://minnie.tuhs.org/TUHS/)
- PDP Unix Preservation Society (http://minnie.tuhs.org/PUPS/)
- The Computer History Simulation Project (http://www.tiac.net/users/mps/retro/)
- http://museum.bablo.ru/ftp/PDP/
- APPENDIX #4 - A short history of UNIX (http://www.ee.ic.ac.uk/docs/software/unix/begin/appendix/history.html)
- UNIX for the first time (http://www.ee.ic.ac.uk/docs/software/unix/begin/)
- 1.3 History of UNIX (http://www.busan.edu/~nic/networking/puis/ch01_03.htm)
- odkaz ()

2.1. Kořeny

Odkazy:

- History of Unix (http://www.melbournelinux.com/unix_history.html)

Kořeny UNIXu se dají vysledovat do projektu Multics. Projekt Multics započal někdy v roce 1965. Podílely se na něm AT&T Bell Laboratories, MIT a General Electric. Cílem projektu bylo vyvinout nový operační systém, jenž měl být multiuživatelský, měl být schopen běhu na více procesorech a měl mít hierarchický systém souborů. V roce 1969 odstoupila Bell Labs od projektu pro nespokojenost s rychlostí vývoje, jenž byl příliš pomalý.

Brzy po opuštění projektu, zahájili někteří z programátorů jenž se na něm podíleli, práci na svém vlastním novém operačním systému. Hlavními programátory byli Ken Thomson a *Dennis Ritchie*. Na prvních pracech kolem UNIXu se podíleli také: R.H. Canaday, M.D. McIlroy a J.F. Ossanna.

V roce 1969 měli Thomson, *Dennis Ritchie* a *Dennis Ritchie* v poznámkách hotovu myšlenku souborového systému. V té době psal Thomson hru „Space Travel“, což byla simulace kosmické lodi na cestě sluneční soustavou. Neměl však pro toto hru vhodný počítač. Pak našel nepoužívaný PDP-7 od firmy Digital Equipment Corporation (../electronic/DEC.html). Operační systém tohoto počítače mu však nevyhovoval a proto se rozhodl napsat vlastní. Začal implementací souborového systému který před časem navrhl. Brzy s k němu připojil *Dennis Ritchie* a společne napsali základní programové vybavení, shell, file copy, delete, edit. *Brian Kernighan*, další člověk z opuštěného projektu Multics, v legraci navrh pro Thomsonův/Ritchie systém jméno "UNICS".

Kapitola 3. Linux

Distribuce Linuxu

- Debian
- Ubuntu
- Red Hat
- Xandros
- Yellow Dog
- …

3.1. Debian GNU/Linux

Kapitola 4. Minix

Odkazy:

- MINIX 3 (http://www.minix3.org/)
- MINIX.ORG (http://www.minix.org/)

Kapitola 5. BSD Unix

Odkazy:

- NET BSD
- Free BSD
- Open BSD

II. Správa UNIXu

Kapitola 6. Instalace a základní konfigurace Debian/GNU Linuxu

* *chapter id="instalace-linux-debian"*
* *print="**psselect -p49-97 unix.ps**|**foldprn -s52**"*

V této kapitole se věnuji instalaci a prvotní konfiguraci Linuxu. Probírám jednotlivé způsoby instalace a uvádím větší množství různých příkladů pro ozřejmení.

* *Tuto kapitolu je třeba přeorganizovat a částečně přepsat. Je velmi zmatečná.*

Návrh organizace kapitoly:

1. Instalace Woody
2. Instalace Sarge
3. Instalace na ZIP
4. Duplikování instalace
5. české prostředí
6. ...
7. FAI

Ostatní sekce dojdou rozhodit k jiným kapitolám, či do samostatné kapitoly.

6.1. Plánování instalce

* *section id="installation-planning" xreflabel="Plánování instalace"*

Před vlastním zahájením instalace je třeba provést pečlivé naplánování. U jednodušších isntalací může zkušený administrátor tuto část „přeskočit“. To znamená že ji provede v paměti těsně před instalací. U složitějších instalací a instalací serverů se vyplatí provést řádné plánování. Co se teda rozumí pod pojmem plánování instalace. Jsou to dvě akce:

- sběr informací
- přidělování zdrojů

Sběr informací. Pod tímto pojmem rozumíme shromažďování informací o určení stanice/serveru. Zajímá nás:

- k jakému účelu bude používána
- jaké bude zatížení, tedy jak veliký bude objem zpracovávaných dat od kterého se odvíjí další tři body
- jaké nároky budou kladeny a úložný diskový prostor
- jaké nároky budou kladeny na paměťový prostor pracovní RAM paměti stroje
- jaké nároky budou kladeny na výkon procesoru
- **FIXME:**doplnit.

6.1.1. Rozdělení disků

(06.05.2005 11:17:46) Marvin:
rpozdělení disku na oddíly je docela fuška. vychází se přitom z plánovaného nasazení a zkušeností.
namountovat lze cokoliv kamkoliv. připojování oddílů se řídí informacemi v /etc/fstab

A jhak to mám tedy udělat? Vše na jednom oddílu?

(06.05.2005 11:24:01) Marvin:
ne, má se vyčlenit oddíl pro /, další pro /usr, další pro /var, další pro /home a pak podle potřeby. Vychází se z plánovaného nasazení stroje kdy jsou konkrétní představy o objemové náročnosti v jednotlivých částech adresářové struktury.

(06.05.2005 11:24:29) Marvin:
má to řadu důvodů

(06.05.2005 11:27:18) Marvin:
takže nejdříve se určí nároky na swap. platí základní omezení na velikost swap oddílu. Linux využije nejvýše 2GB z oddílu takže nemá smysl dělat větší to se pak udělá více swap oddílů.

kolik je plánovaná poteba RAM pro všechny porcesy?
tj na unixu platí, raději pomalejší procesor a více paměti než naopak.
resp tolik paměti aby se nešáhlo na swap.

takže podle poředpokládaného užití si něco vyhradíš pro swap. jakmile do něj stroj šáhne mělo by se přehodnotit paměťové nároky a opět zvážit změny.

pak se určí oddíl pro root ten by měl mít cca 100MB, možná bych dal trochu víc.

takhle, já zvyknu odhadnout potřeby, horní limit ale pak ve skutečnosti nastavím dvakrát tolik kolik.

(06.05.2005 11:43:46) Marvin:
no takhle 2GB na čistý root je zbytečně mnoho, třeba na jednom server tam mám obsazeno míň než 50MB. Ale je taky pravda že jsem docela nevychovaně dříve s / dával do stejného oddílu taky /usr a /tmp a proto jsem dával na / tak 1-3GB.

ext2 nemá journal. tzn že není vhodný pro oddíly větší než několik GB. je tam problém s fsck který na nežurnálových systémech trvá dlouho. dle kapacity disku by to mohlo být i několik desítek minut.

Rozdělení disku:

```
partition   filesystem          mount point     space
----------  ------------------  --------------  --------
/dev/hda1   swap                SWAP
/dev/hda2   ext2/3              /               200MB
/dev/hda3   ext3/reiser         /tmp            >1GB
/dev/hda4   EXTENDED
/dev/hda5   ext2/3              /usr            0.5-2GB
/dev/hda6   ext3/xfs            /var            >1GB
/dev/hda7   ext3/xfs            /home           podle uživatelů
/dev/hda8                       /opt            0 - rezerva
zbytek disku nepoužit a ponechá pro budoucí rozšíření
```

6.2. Instalace Debian GNU/Linux 3.x Woody

* *section id="install-woody"*
* *print="**psselect -p50-56 unix.ps|foldprn -s8**"*

Instalace Debian GNU/Linux verze Woody.

Následující postup nemusí být úplným. Je to jen pokus zaznamenat průběch instalace.

- Choose The Language: **`en -- Choose this and press Enter to proceed in English`**
- Choose Language Variant: **`English (United States)`**
- Configure the Keyboard
  - Select a keyboard: **`qwerty/cz-lat2: Czech`**
- Partition a Hard Disk
- Install Kernel and Driver Modules
- Configure Device Driver Modules

6.2.1. Příprava instalace

* *print="**psselect -p50-51 unix.ps|foldprn -s4**"*
* *Zde uvedu seznam věcí/nastavení které potřebujeme znát v průběhu instalace.*

Budeme-li instalovat ze sítě, potřebujeme znát server z kterého se bude instalace provádět.

V případe že používám apt-proxy uvedu jako server `debian:9999`. `debian` je alias který jsem pro server s apt-proxy zavedl v DNS a `9999` je port na které běží. Dále cestu k distribuci, ta je `main`. Tyto informace se zadávají odděleně.

Rovněž si připravíme konfiguraci pro **apt-get**. Vypíšeme si seznam zdrojů které budeme používat. V případě naší apt-proxy to budou následující (větve uvedené mezi „[“ a „]“ jsou nepovinné):

```
deb http://debian:9999/main woody main [contrib] [non-free]
deb http://debian:9999/non-US woody/non-US main [contrib] [non-free]
deb http://debian:9999/security woody/updates main [contrib] [non-free]
```

6.2.2. Standardní běžná instalace Debian/GNU Linux

* *print="**psselect -p50-56 unix.ps|foldprn -s8**"*

Nejdříve si připravíme instalační diskety. Pod UNIXem se to dělá například takto:

Příklad 6-1. Příprava instalačních disket *Debian Potato*

```
# cd adresář_s_obrazy_disket
# dd if=rescue.bin of=/dev/fd0 bs=1024 conv=sync; sync
# dd if=root.bin of=/dev/fd0 bs=1024 conv=sync; sync
# dd if=driver-1.bin of=/dev/fd0 bs=1024 conv=sync; sync
# dd if=driver-2.bin of=/dev/fd0 bs=1024 conv=sync; sync
# dd if=driver-3.bin of=/dev/fd0 bs=1024 conv=sync; sync
# dd if=driver-4.bin of=/dev/fd0 bs=1024 conv=sync; sync
```

* ***FIXME:**Ověřit nutnost použití parametru conv=sync*

Po před každým příkazem **dd** připravíme do první disketové mechaniky `/dev/fd0` (`A:`) disketu na kterou bude uložen obraz z příslušného souboru. Diskekty si pečlivě označíme, protože v průběhu instalace si o ně bude instalační program říkat.

Adresář s obrazy disket se liší podle verze distribuce. Pro Debian GNU/Linux jsou to tyto adresáře:

```
.../debian/dists/potato/main/disks-i386/current/images-1.44
.../debian/dists/woody/main/disks-i386/current/images-1.44
```

Rovněž počet obrazů disket s ovladači, diskety `driver-?` se může lišit.

6.2.2.1. Instalace jádra z disket

Zasuneme disketu označenou jako `rescue` do disketové mechaniky a „nabootujeme“ z ní. Poté si počítač řekne o disktu označenou jako `root`, na této je souborový systém s programy potřebnými pro instalaci.

- Configure the Keyboard: qwerty/cz_lat2 (Czech)
- Preload essential modules from a floppy

6.2.2.2. Instalace jádra z disket Woody 1.44 Compact

Instalace je obdobná jako z základních 1.44 disket. Tato varianta je zjednodušená o řadu ovladačů a jsou k ní přidány ovladače Mylex Acceleraid řadiče a další.

- Configure the Keyboard: qwerty/cz_lat2 (Czech)
- Partition a Hard Disk
- Initialize and Activate a Swap Partition
- Initialize a Linux Partition - zinicializuji jen `/` a `/boot`
- Install Kernel and Driver Modules
- Configure the Network
    - Name: moon
    - Choose the IP Address: **`10.16.66.20`**
    - Choose the Network Mask: **`255.255.224.0`**
    - What is your IP gateway address?: **`10.16.66.50`**
    - Choose the Domain name: **`moraviapress.cz`**
    - Choose the DNS Server Addresses: **`10.16.66.18 10.16.66.19 10.16.66.20`**
- Install the Base System
    - Select Installation Medium: network (HTTP or FTP over the network)
    - Select Installation Server
        - Download URL: http://debian:9999/main
        - Proxy: NO
- Make System Bootable
    - Where should the LILO boot Loader be installed: /dev/rd/c0d0
- Make a Boot Floppy
- Reboot the System

Po restartu počítače tento nabootuje z disku. Nenabootuje-li z disku, můžeme zkusit záchrannou disketu kterou jsme si právě vytvořili.

Spustí se konfigurační skript `/usr/sbin/base-config`

6.2.2.3. Instalace programů

Instalaci programů provádíme příkazy **dpkg**, **dselect**, **apt-get** či **wajig**. Uvedl jsem jen konzolové programy z nichž všechny mimo **dselect**, jenž je „celoobrazovkový“, jsou řádkové utility.

* ***FIXME:*** *dopsat pár slov o* ***dpkg****.*

Nastavení konfigurace **apt** se provádí konfiguračními soubory v adresáři `/etc/apt/`. Jedním z nich je soubor `/etc/apt/sources.list` jenž popisuje odkud **apt** získá balíčky s programy.

Příklad 6-2. Ukázkové nastavení tohoto souboru

```
# /etc/apt/sources.list

# Debian potato in local archive
deb ftp://debian/debian potato main contrib non-free
deb ftp://debian-non-US potato/non-US main contrib non-free

# Last security updates
deb http://security.debian.org potato/updates main contrib non-free
```

Příklad 6-3. Ukázka `/etc/apt/sources.list` pro distribuci Woody

```
# Debian potato in local archive
deb ftp://debian/debian woody main contrib non-free
deb ftp://debian-non-US woody/non-US main contrib non-free

# Last security updates
deb http://security.debian.org woody/updates main contrib non-free
```

Příklad 6-4. Ukázka `/etc/apt/sources.list` pro stabilní distribuci a české archívy

```
# Debian 'stable' in Czech archive
deb http://www.debian.cz/debian stable main contrib non-free
deb http://www.debian.cz/debian-non-US stable main contrib non-free

# Last security updates
deb http://security.debian.org stable/updates main contrib non-free
```

Příklad 6-5. Ukázka `/etc/apt/sources.list` pro lokální „cache“ balíčků

```
# Debian cache
deb http://debian:9999/main woody main contrib non-free
deb http://debian:9999/non-US woody/non-US main contrib non-free

# Last security updates
deb http://security.debian.org stable/updates main contrib non-free
```

6.2.2.4. Instalace Woody ze sítě

Select Installation Server

Instalaci provedu ze serveru www.debian.cz (http://www.debian.cz)

```
Please provide the URL from witch to download the installation files.
```

```
If you have a proxy server, fill that out as well; otherwise, leave it
set  to 'none'

Download URL http://www.debian.cz/debian/dists/woody/main/disks-i386/current/

Proxy        none                                        Proxy Port 8080
```

6.2.3. Instalace z neoficiálního miniCD

Další možností, kterou máme k dispozici je neoficiální MiniCD.

Po nabootování z CD jsme v menu kde si můžeme přečíst základní nápovědu a poté zvolit jaké jádro a s jakými parametry chceme zavést. Já v dnešní době používám hodně jádro bf24 protože má v sobě podporu pro ReiserFS který s oblibou na velikých discích nasazuji. Ale můžeme si vybrat z jader:

- `linux` nebo `idepci`
- `bf24`
- `compact`
- `vanilla`
- `rescue`, `rescbf24`, `resccomp` nebo `rescvanl`

Podle potřeby můžeme za jádro zapsat parametry

- `root=/dev/fd1`
- `mono` — pro monochromatické monitory
- `video=vga16:off` — zakáže framebuffer pro monitor
- `hd=cylinders,heads,sectors` —
- `floppy=thinkpad` — IBM ThinkPad
- `no387` — Vypíná FPU na starších strojích
- `apm=on` — Advanced Power Management
- `aha152x=iobase[,irq[,scsi-id[,reconnect]]]` — Advanced Power Management

Speciální bootovací parametry:

- `quiet` — Tichý mód, mín se vyptává
- `verbose` — Ukecaný mód
- `debug` — Ladicí mód, ladicí informace jsou na `tty3`
- `bootkbd` — Nastaví překlad pro klávesnici, například `bootkbd=qwerty/us`

Zavedeme tedy jádro bez dalších parametrů.

```
boot: bf24
```

Po zavedení jádra se spustí instalační a konfigurační skript. Tento začíná volbou jazyka, Choose The Language⟶cs - Volíte tento a Enter k pokračování česky a varianty jazyka Zvolte variantu jazyka⟶Česky V hlavní nabídce pak zvolíme další volbu Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Konfigurovat klávesnici a nakonfigurujeme si anglickou klávesnici. Vybrat klávesnici⟶qwerty/us: Anglicky U.S. (QWERTY) S českou klávesnicí na konzole nemám dost zkušeností a proto pro jistotu volím anglickou.

Poté je nám jako další menu nabídnuto rozdělení pevného disku, pokud již není rozdělen. Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Rozdělit pevný disk Postupně si vybereme disky které

chceme rozdělit a používat v menu Vybrat diskovou jednotku Spustí se nám program **cfdisk** pro práci s tabulkou rozdělení disku. V něm vytvoříme příslušné oddíly které jsme si předem naplánovali.

* ***FIXME:****napsat článek o plánování rozdělení disku a sem umístnit odkaz.*
Na disku na serveru (RAID) který má cca 109GB jsem použil následující rozdělení

disk	velikost	fs	mount point
/dev/sda1	3GB	swap	
/dev/sda2	1GB	ext2	`/`
/dev/sda3	5GB	reiserfs	`/var`
/dev/sda5	30GB	reiserfs	`/home`
/dev/sda6	70GB	reiserfs	`/home/osvit`

Po rozdělení disku je na čase jednotlivé oddíly inicializovat a připojit. Jako první je volba Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Inicializovat a aktivovat odkládací oddíl Poté postupně inicializujeme ostatní oddíly a připojíme. Vyjdeme přitom z tabulky kterou jsme si předem připravili. Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Inicializovat linuxový oddíl

Nyní můžeme přistoupit k instalaci jádra a modulů Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Instalovat jádro a moduly operačního systému Instalační skript nám ukáže připojené svazky/oddíly na které bude instalovat

Obrázek 6-1. Ověření výběru souborových systémů

```
Připojeny jsou následující souborové systémy, na které se
nainstaluje jadro, moduly a základní systém:
/dev/sda6 připojen na /home/osvit
/dev/sda5 připojen na /home
/dev/sda3 připojen na /var
/dev/sda2 připojen na /
```

Vzhledem k tomu že základní systém je na instalačním MiniCD, nechám proběhnout instalaci z něj.

Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Konfigurovat moduly s ovladači zařízení Z modulů potřebuji akorát ovladač síťové karty, která se po startu nenašla. `kernel/drivers/net`: `eepro100` Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Konfigurovat síť Vybrat hostitelský název⟶sunset2

Hlavní nabídka instalace systému Debian GNU/Linux⟶Další: Instalovat základní systém

Po restartu se spustí `Debian System Configuration`

```
If you want to revisit this setup process at a later data, just run
/usr/sbin/base-config
```

6.2.4. Výměna jádra 2.2.x za 2.4.x

Pokud potřebujeme některé vlastnosti nového jádra, můžeme nahradit standardní jádro za některé z novějších. Takže, jestli jsme přímo z instalačního média neinstalovaji jádro `bf24`, je teď čas nějaké 2.4.x jádro vybrat. Vybírejte pečlivě, jádra jsou optimalizována pro konkrétní procesory. Máme-li jádro vybráno můžeme instalovat.

```
# apt-get innstall kernel-image-2.4.verze-architektura
```

V průběhu instalace budeme upozorněni na **nutné** úpravy v konfiguraci systému. Protože jádra řady 2.4 jsou přeložena s podporou ramdisku a s základními ovladači jako moduly na ramdisku je třeba toto zohlednit v konfiguraci použitého zavaděče. Pro případ lila se jedná o vložení řádku

```
initrd=/initrd.img
```

do standardní sekce nového/aktuálního jádra

```
image=/vmlinuz
```

Tedy pokud nemáme konfiguraci upravenou jinak. Po opravě konfigurace musíme tuto zapsat na disk do bootovací oblasti příkazem

```
# lilo -v -v
```

Ty přepínače `-v` jsou tam abychom byli informování o postupu práce lila. Tím máme počítač připraven k natažení nového jádra. To na co nesmíme zapomenout jsou drobné rozdíly mezi moduly (jejich názvy) u jednotlivých řad linuxových jader. Tedy po restartu počítače musíme dokonfigurovat tyto moduly. Pokud nemáme přesný popis použté techniky, můžeme si vypomoci prozkoumáním deníku `/var/log/syslog`. Zde je popsána detekovaná technika při startu počítače. Dalším možnám způsobem je příkaz **lspci**.

6.2.5. Poinstalační konfigurace

Tato část je více osobní. Popisuji zde zásahy do instalace a změny kterými upravuji instalovaný systém k „obrazu svému“.

První věcí kterou provedu je rekonfigurace alternativ (`/etc/alternatives/`). První věcí kterou změním je standardní editor, tím je totiž po instalaci **nano** který mi nevyhovuje. Nastavím tedy **vi**.

```
# update-alternatives --config editor

There are 3 programs which provide ‘editor’.

  Selection    Command
-----------------------------------------------
*+    1        /usr/bin/nano
      2        /bin/ed
      3        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 3
Using ‘/usr/bin/nvi’ to provide ‘editor’.
```

Pokud máte další zvyklosti můžet zde nastavit alternativy podle svého. V této chvíli možností mnoho není, neb je nainstalován jen základní software. Dalším krokem tedy bude doinstalování programů na které jsme zvyklí. Já instaluji tyto: `less`

- **less** — jako alternativa/náhrada k **more**
- **ssh** — pro vzdálenou správu, rovněž povolím spuštění **sshd** serveru/démona
- `iproute` — obsahuje program **ip** pro práci se směrovacími tabulkami, a další. Alternativa/náhrada za **route**.

```
# apt-get install less ssh iproute
```

6.2.6. Poznámky k instalaci

Po restartu počítače v průběhu instalace když je již natažen základ se spustí skript `/usr/sbin/base-config` tento nás provede dokončení konfigurace. Skript se dá spusti i později.

6.3. Instalace Debian GNU/Linux 3.1 Sarge

* *section id="install-sarge" condition="author"*
* ***FIXME:****zde bude popis standardní instalace Sarge*

Odkazy:

- The Perfect Setup - Debian Sarge (3.1) (http://www.howtoforge.com/perfect_setup_debian_sarge)

6.3.1. Instalace s pomocí disket

Nejdříve si připravíme/stáhneme obrazy disket, například z us.debian.org (http://http.us.debian.org/debian/dists/sarge/main/installer-i386/current/images/floppy/).

```
# wget http://http.us.debian.org/debian/dists/sarge/main/installer-i386/current/images/floppy/
# wget http://http.us.debian.org/debian/dists/sarge/main/installer-i386/current/images/floppy/
# wget http://http.us.debian.org/debian/dists/sarge/main/installer-i386/current/images/floppy/
```

Stažené obrazy zapíšeme na diskety

```
# dd if=boot.img of=/dev/fd0
# dd if=root.img of=/dev/fd0
# dd if=net-drivers.img of=/dev/fd0
```

6.3.2. Dokončení instalace

V rámci dokončení instalace provedeme úpravy konfigurace a doinstalování potřebných balíčků.

Opravíme fatální nedostatky standardní konfigurace aptitude. Vytvoříme soubor `/etc/apt/apt.conf` a do něj vložíme

```
// Konfigurace Aptitude
Aptitude {
        Recommends-Important false;     // Neinstaluj doporucene balicky
}
```

6.4. Instalace Debian GNU/Linux 4.0r3 Etch

Odkazy:

- Distribuční adresář s torrenty pro obrazy CD Debian 4.0r3 Etch (http://cdimage.debian.org/debian-cd/4.0_r3/i386/bt-cd/)
- The Perfect Setup - Debian Etch (4.0) (http://www.howtoforge.com/perfect_setup_debian_etch)

V archivu Debianu je spousta obrazů médií. Jak CD tak i DVD. Pokud budeme instalovat ze sítě, stačí nám vybrané obrazy CD médií. K dispozici jsou tyto:

- `debian-40r3-i386-netinst.iso` — síťová instalace, obraz má asi 160MB
- `debian-40r3-i386-kde-CD-1.iso` — instalace s předpřipraveným KDE
- `debian-40r3-i386-xfce-CD-1.iso` — instalace s předpřipraveným Xfce

Jak poznámky napovídají, pokud budeme chtít instalovat bez grafického prostředí, obvykle server, použijeme první obraz `netinst`. Pokud budeme chtít nainstalovat předpřipravené KDE nebo Xfce, použijeme druhé dva obrazy. Obrazů je v distribučním adresáři více a můžeme zkusit použít jiný, ale já jsem téměř vždy prováděl síťovou instalaci s pomocí obrazu `netisnt`.

Stáhneme si tedy obrazy které jsme si vybrali. V případě torrentu například takto:

```
$ btdownloadcurses http://cdimage.debian.org/debian-cd/4.0_r3/i386/bt-cd/debian-40r3-i386-neti
```

A nahrajeme na volná CD-R nebo CD-RW média, například takto:

```
$ wodim speed=4 -v -eject -data debian-40r3-i386-netinst.iso.torrent
```

Nahrané médium vložíme do počítače a zavedem z něj systém. Po zavedení systému se objeví obrazovka s výzvou. Nyní můžeme zmáčknou enter a spustit bežnou instalaci `install`, nebo zadat jiný typ instalace z množiny dostupných. Jsou to například:

- `install`
- `expert`
- `installgui`
- `expertgui`
- `rescue`
- `rescuegui`

6.5. Instalace Debian GNU/Linux 5.0 Lenny

Obrazy instalčních médií si můžem stáhnout například z cdimage.debian.org (http://cdimage.debian.org/debian-cd/).

6.6. Instalace Debian GNU/Linux 6.0 Squeeze

6.7. FAI (*Full Automatic Installer*)

* *section id="fai" xreflabel="FAI"*

Popis konfigurace a použití systému plně automatické instalace FAI. Tato část není úplná. Po nějaké době hraní si s FAI jsem toto odložil neb jsem jej nutně nepotřeboval. Někdy s k tomuto tématu opět navrátím abych jej dokončil.

Zdroje a odkazy:

- FAI - Full Automatic Installer I (http://www.root.cz/print.php4?id=2118)
- FAI - Full Automatic Installer II (http://www.root.cz/print.php4?id=2149)
- FAI - Full Automatic Installer III (http://www.root.cz/print.php4?id=2174)
- FAI - Full Automatic Installer IV (http://www.root.cz/print.php4?id=2198)

6.7.1. Instalace

Nejdříve jsem na serveru `moon` nainstaloval fai

```
moon:~# apt-get install fai
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  debootstrap
The following NEW packages will be installed:
  debootstrap fai
0 packages upgraded, 2 newly installed, 0 to remove and 0  not upgraded.
Need to get 457kB of archives. After unpacking 1630kB will be used.
Do you want to continue? [Y/n]
Get:1 http://debian woody/main debootstrap 0.1.17 [52.8kB]
Get:2 http://debian woody/main fai 2.3.1 [404kB]
Fetched 457kB in 28s (16.2kB/s)
Instaluji balík debootstrap.
(Čtu databázi ... nyní je nainstalováno 21053 souborů a adresářů.)
Rozbaluji debootstrap (z .../debootstrap_0.1.17_i386.deb) ...
Instaluji balík fai.
Rozbaluji fai (z .../apt/archives/fai_2.3.1_all.deb) ...
Nastavuji balík debootstrap (0.1.17) ...

Nastavuji balík fai (2.3.1) ...
To set up FAI, edit /etc/fai/fai.conf and call fai-setup.
```

poté jsem nainstalova jádra

```
moon:~# apt-get install fai-kernels
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  fai-kernels
0 packages upgraded, 1 newly installed, 0 to remove and 0  not upgraded.
Need to get 11.3MB of archives. After unpacking 11.4MB will be used.
Get:1 http://debian woody/main fai-kernels 1.2 [11.3MB]
Fetched 11.3MB in 11m15s (16.7kB/s)
Instaluji balík fai-kernels.
(Čtu databázi ... nyní je nainstalováno 21295 souborů a adresářů.)
Rozbaluji fai-kernels (z .../fai-kernels_1.2_i386.deb) ...
Nastavuji balík fai-kernels (1.2) ...
```

Doinstaloval jsem ještě řadu balíčků

- `mknbi` —
- `tftpd-hpa` — tftp démon s PXE boot protokolem

Opravil konfiguraci v souboru `/etc/fai/fai.conf`. Oprava se týkala těchto nastavení:

```
FAI_DEBOOTSTRAP="$debdist http://debian:9999/main"

FAI_SOURCES_LIST="deb http://debian:9999/main $debdist main contrib non-free
deb http://debian:9999/non-US $debdist/non-US main contrib non-free
deb http://security.debian.org/ $debdist/updates main contrib non-free"

KERNELPACKAGE=/usr/lib/fai/kernel/kernel-image-2.2.20_DHCP1_i386.deb
```

Ostatní nastavení jsem nechal na původních hodnotách. Poté jsem konečne spustil fai-setup

```
# LANG=C fai-setup
```

lépe je však zachytit výstup tohoto skriptu pro pozdější podrobné studium. To učiníme takto.

```
# LANG=C fai-setup 2>&1 |tee fai-setup.log
```

Vytvoření bootovací diskety:

```
# make-fai-bootfloppy "nfsroot=/usr/lib/fai/nfsroot nfsaddrs=kernel FAI_FLAGS=sshd"
Creating filesystem on floppy device /dev/fd0.
mke2fs 1.27 (8-Mar-2002)
Creating the boot floppy with grub.
Using IP address 10.16.66.20 of eth0 for the fixed boot menu.
Writing boot data to floppy. The default boot label is:
The kernel configuration is /usr/lib/fai/nfsroot/boot/config-2.4.20-fai.
Additional kernel parameters: nfsroot=/usr/lib/fai/nfsroot nfsaddrs=kernel FAI_FLAGS=sshd
```

Chceme-li použít jiné jádro, například námi překládané, upravíme `/etc/fai/fai.conf`. Vepíšeme do něj, nebo přepíšeme řádek

KERNELPACKAGE=/usr/usr/kernel-image-2.4.20-fai_0_i386.deb

Poznámka: Ukázka překladu jádra na verzi `fai-dhcp_2`

```
# make-kpkg clean
# make-kpkg --append_to_version -fai-dhcp --revision 2 \
              kernel_image modules_image
```

> **Varování**
>
> Do jádra musí být zabudována podpora pro nfs klienta a všechen hardware nutný k instalaci.

Do konfiguračního souboru jsem pro jádro 2.4.x přidal

FAI_LOCATION="moon:/usr/local/share/fai"

Stanice kterou budu přes fai instalovat je DELL OptiPlex GX1 a má MAC adresu `00:c0:4f:a7:76:b7`

* *Možná bude třeba doinstalovat bootp / dhcp / dhcp3.*

Problémy. Listing končí takto:

```
Creating SSH2 RSA key
Creating SSH2 DSA key
Restarting OpenBSD Secure Shell server: sshd.
/vmlinuz does not exist. Installing from scratch, eh?
```

```
Or maybe you don't want a symbolic link here. Hmm? Lets See.
I notice that you do not have vmlinuz symbolic
link. I can create one for you, and it shall be
updated by newer kernel image packages. This is
useful if you use a boot loader like lilo.
Do you want me to create a link from /boot/vmlinuz-2.4.18-bf2.4 to vmlinuz?[Yn] Unknown optior
/boot/fai/installimage: file not found
BOOTP environment prepared.
make-fai-nfsroot finished.
Stopping NFS kernel daemon: mountd nfsd.
Unexporting directories for NFS kernel daemon...done.
Exporting directories for NFS kernel daemon...done.
Starting NFS kernel daemon: nfsd mountd.
FAI setup finished.
moon:~#

Saving log files remote to fai@ fai0001/sysinfo-20030310_110335
rsmd: getaddrinfo: Name or service not known
rcmd: getaddrinfo: Name or service not known
Press <RETURN> to reboot or ctrl-c to execute a shell
```

6.7.2. Adresáře a soubory

V adresáři `/etc/fai` se nacházejí dva soubory:

`/etc/fai/fai.conf`

: Základní konfigurace fai ze které se připravuje adresářová struktura FIXME:. Po nakonfigurování se spustí program `fai-setup`

`/etc/fai/sources.list`

: Tento soubor se uplatní, není-li v `fai.conf` definována proměnná `FAI_SOURCES_LIST`.

`/usr/lib/fai`

: adresář

  Obsahuje jako podadresář `nfsroot` který si klienti při instalaci připojují jako kořenový adresář.

`/usr/local/share/fai`

: Adresář s konfiguracemi. Tento si připojují instalované stanice jako `/fai`.

`/usr/lib/fai/kernel`

: V tomto adresáři jsou uložena jádra která používáme během instalace klientů a která na tyto klienty instalujeme. Jádro které se zavádí ze sítě je určeno parametrem *`KERNELPACKAGE`* v souboru `/etc/fai/fai.conf`.

`/usr/local/share/fai`

: V tomto adresáři jsou uloženy konfigurace a konfigurační skripty. Tento adresář si v době instalace připojují klienti jako `/fai`.

6.7.3. Konfigurace DHCP

6.7.3.1. dhcpd3

Konfigurace pro `dhcpd3` (sarge, 3.0+3.0.1rc9-5)

```
option option-170 code 170 = string;
option option-171 code 171 = string;
option option-172 code 172 = string;
option option-173 code 173 = string;
option option-174 code 174 = string;

# FAI Setup
filename "/boot/fai/pxelinux.0";
option dhcp-max-message-size 2048;
use-host-decl-names on;
option root-path "/var/data/fairoot";
option option-170 "shining:/usr/local/share/fai";       # FAI_LOCATION
option option-171 "sysinfo";                            # FAI_ACTION
option option-172 "verbose createvt sshd";              # FAI_FLAGS
server-name "server";
```

6.7.4. Prvotní konfigurace

Prvotní konfigurace sestává se správného nastavení v souboru `/etc/fai/fai.conf`, připravení jádra a případného PXE, vytvoření bootovací diskety, doinstalování nezbytného softwaru.

6.7.4.1. Soubor `fai.conf`

Vypíšu položky jejichž význam znám.

`installserver=moon`

Instalační server. Tedy server který v průběhu instalace klienti používají.

`debdist=woody` # distribution: woody, sarge, sid

Název verze která se bude instalovat. Tento parametr je možné pro vybrané stanice změniv v konfiguraci. FIXME: popsat kde.

`FAI_FLAGS="sshd"`

Příznaky jejichž nastavení ovlivní chování FAI.

Tabulka 6-1. FAI_FLAGS

`createvt`	na instalovaném klientovi je možno otevřít sezení
`debug`	FIXME: doplnit
`sshd`	FIXME: doplnit
`verbose`	FIXME: doplnit

Tento parametr můžeme také nastavit přess `append`

```
append ip=dhcp root=/dev/fs FAI_FLAGS=verbose,createvt,sshd
```

například při zavádění systému přes PXE je tento append v souboru `/boot/fai/pxelinux.cfg/default`

`debug=true`

Touto proměnnou zapneme „ladění“, tedy ve výpisech budou informace navíc. Použijeme při ladění skriptů, nebo pro podrobnější informace máme li problém a něco nefunguje.

6.7.5. Konfigurační adresář `/fai`

V podadresářích tohoto adresáře jsou uloženy konfigurační skripty které se spouštějí na klientovi v době instalace. Nachází se na instalačním serveru v adresáři `/usr/local/share/fai`, je vyexportovaný přes nfs a klienti si jej připojují do adresáře `/fai`.

V podadresáři `class` jsou uloženy popisy tříd a skripty které přiřazují třídy jednotlivým instalovaným stanicím.

Významy některých souborů

`LAST.var`

V tomto souboru se určuje jaká akce se se stanicí provede. Implicitní akce je definována na konci souboru v řádku

```
[ -z "$FAI_ACTION" ] && FAI_ACTION=sysinfo
```

V souboru ve funkci `mktable` je uveden seznam stanic následovaný akcí. Protože akce `sysinfo` je implicitní nemusíme ji zde uvádět.

Standardně jsou definovány dvě akce `sysinfo` která zjišťuje systémové informace o stanici a výsledek zjištění zapíše do adresáře `/home/fai/`*`jméno_stanice`*`/last-sysinfo`. Druhou akcí je `install` jenž nainstaluje stanici podle specifikované konfigurace. Konfigurace je dána dalšími konfiguračními skripty.

Akce

Je možné dodefinovat si další akce.

`01alias`

Zde přidělujeme třídy jednotlivým stanicím. Soubor je vykonavatelný skript jenž do stdout zapíše jména tříd které se na danou stanici vztahují. Například já zde mám uvedeno

```
case $HOSTNAME in
    eye??) #our eyes (web machines)
        cat eye
        ;;
    xtrm??) # X-Terminals
        echo xtrm
        ;;
esac
```

`xtrm` — vlastní soubor

V tomto souboru uvádím seznam softwaru (tříd) které se mají na dané stanici nainstalovat. Například v tomto konkrétním je uvedeno:

```
GRUB
BASE
```

```
NETWORK NTP
```

6.7.5.1. Některé třídy a jejich použití

`BOOT`

Třída počítačů které zavádějí systém. Příslušnost k této třídě způsobí instalaci zavaděče na lokální disk (Aspoň si to myslím). V určité fázi instalace dojde ke spuštění skriptu `/fai/scripts/BOOT`. Tato třída sama nestačí ještě je třeba nainstalovat některý ze zavaděčů systém příslušností ke třídě `LILO` nebo `GRUB`

6.8. Alternativy

V adresáři `/etc/alternatives` jsou alternativy k programům. Co se tím myslí. Například nastavíme standardní editor na `nvi` a všechny programy jenž používají program `editor` pak budou spouštět `nvi`

Nastavení provedeme spuštěním programu

```
# update-alternatives --configure editor
```

a z menu si vybereme který program chceme nastavit jako standardní editor.

6.9. Záplatování a kompilace jádra

Záplatování a kompilaci si předvádím na jádru 2.4.18 kompilovaném pod a pro Debian GNU/Linux Woody. Nejdříve si nainstalujeme zdroje jádra a potřebné programy ke kompilaci.

```
# wajig install kernel-source-2.4.18
```

zdroje rozbalíme

```
# cd /usr/src
# bzip2 -cd kernel-source-2.4.18.tar.bz2 |tar xvf -
# ln -s kernel-source-2.4.18 linux
```

doinstalujeme potřebné balíčky

```
# wajig install ncurses-dev kernel-package
```

6.9.1. Konfigurace

```
# cd /usr/src/linux
# make menuconfig
```

6.9.2. Kompilace

Odkazy, zdroje:

- http://www.debian.org/doc/manuals/reference/ch-kernel.en.html

- http://www.myrddin.org/howto/debian-kernel-recompile.html

```
# cd /usr/src/linux
# make-kpkg --revision moon.0 kernel_image
```

Jednoduchý postup

1. Nainstalujeme zdroje jádra

   ```
   # apt-get install kernel-source-2.4.18 kernel-package debhelper dpkg-dev libncurses5-de
   # cd /usr/src
   # tar xjf kernel-source-2.4.18
   # ln -s kernel-source-2.4.18 linux
   ```

2. Nakonfigurujeme

   ```
   # cd /usr/src/linux
   # make menuconfig
   ```

3. A vytvoříme balíčky. Obecně takovýmto postupem

   ```
   # make-kpkg clean
   # make-kpkg --append_to_version -počítač --revision číslo revize \
               kernel_image modules_image
   ```

   Například jádro čtvrté revize pro počítač `futok` se připraví takto

   ```
   # make-kpkg clean
   # make-kpkg --append_to_version -futok --revision 4 \
               kernel_image modules_image
   ```

   Podle toho, používáme-li `initrd`, přidáme volbu `--initrd`

Jádro můžeme nainstalovat pomocí

```
# dpkg -i kernel-image-2.4.18-futok.4.deb
```

6.9.2.1. Nezpracováno

```
# make oldconfig
# make menuconfig
# make-kpkg clean
# make-kpkg --initrd --revision=2:hostname.1.0 kernel_image
```

6.9.3. Překlad jádra se záplatami

* *section id="compile_kernel_with_patches" xreflabel="Překlad jádra se záplatami"*

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

* *print="**psselect -p 36-40 unix.ps|foldprn -s 8**"*

Například zkusíme přeložit jádro se záplatou `ctx` pro virtuální server. Musíme mít samozřejmě nainstalovánu záplatu a nakonfigurován make-kpkg tak aby záplaty používal. To učiníme přidáním řádku

```
patch_the_kernel := YES
```

do souboru `/etc/kernel-pkg.conf`. Poté provedem překlad jádra.

```
# make-kpkg clean
# make-kpkg --append_to_version -futok --revision 4 \
            --added-patches ctx \
            kernel_image modules_image
```

Překlad novějšího jádra bez záplat. Překládám jádro 2.4.22 z *unstable*. Nainstaloval jsem si zdroje jádra

```
# apt-get install -t sid kernel-source-2.4.22
```

Rozbalil a poté se přepnul přímo do rozbaleného adresáře.

* *Nedotkl jsem se symbolického odkazu `/usr/src/linux` který směřuje na záplatované zdroje jádra `2.4.20`.*
Po nakonfigurování jsem jádro přeložil

```
# export PATCH_THE_KERNEL=NO
# make-kpkg clean
# make-kpkg --append-to-version -yoda --revision 1 kernel_image modules_image
```

6.9.3.1. Využití skriptu pro správu záplat

Jedním ze způsobů jak spravovat záplaty jádra je balíček kernel-patch-scripts který se objevil v Sarge. Upravíme si tedy konfiguraci apt abychom mohli nainstalovat tento balíček ze Sarge. Není třeba se obávat, protože je to balíček skriptů bez závislostí na knihovnách ze Sarge.

```
# apt-get install kernel-patch-scripts
```

6.9.3.2. Ukázka překladu jádra

Jedná se o překlad jádra 2.4.22 s moduly pro notebook IBM T30+. Do jádra je zahrnuta formou modulů podpora pro ovládání hardvare notebooku `thinkpad` a podpora šifrovaných tunelů `cipe`, jakožto i podpora pro zvukový čip `alsa-driver` a PC card `pcmcia-cs`. V modulu `cipe` je upraven soubor `cipe/cipe.h`

```
--- cipe/cipe.h.orig    Sat Oct  4 16:44:26 2003
+++ cipe/cipe.h Sat Oct  4 16:50:39 2003
@@ -153,6 +153,9 @@
 #if LINUX_VERSION_CODE >= KERNEL_VERSION(2,3,0)
 #define LINUX_23
 #endif
+#if LINUX_VERSION_CODE == KERNEL_VERSION(2,4,22)
+#define LINUX_26
+#endif
 #if (__GNUC__ > 2 || (__GNUC__ == 2 && __GNUC_MINOR__ >= 91)) && defined(__i386__)
 #define REGPARM /* __attribute__((regparm,3)) XX needs testing */
 #else
```

a soubor `cipe/output.c`

```
--- cipe/output.c.orig  Mon Apr  7 00:53:12 2003
+++ cipe/output.c       Sat Oct  4 17:41:54 2003
@@ -192,10 +192,10 @@
```

```
                             cipe_ntoa(0, dst), cipe_ntoa(1, tunnel->myaddr),
                             RT_TOS(tos), tunnel->sock->bound_dev_if));
 #endif
-#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,7)
+#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,7) && !defined(LINUX_26)
        err = ip_route_output(&rt, dst, tunnel->sock->rcv_saddr, RT_TOS(tos),
                              tunnel->sock->bound_dev_if);
-#elif LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45)
+#elif LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45) && !defined(LINUX_26)
         err = ip_route_output(&rt, dst, inet_sk(tunnel->sock)->rcv_saddr,
                              RT_TOS(tos), tunnel->sock->bound_dev_if);
 #else /* LINUX_VERSION_CODE >= KERNEL_VERSION(2,5,45) */
@@ -204,7 +204,11 @@
               { .oif = tunnel->sock->bound_dev_if,
                 .nl_u = { .ip4_u =
                             { .daddr = dst,
+#if defined(LINUX_26)
+                               .saddr = tunnel->sock->rcv_saddr,
+#else
                               .saddr = inet_sk(tunnel->sock)->rcv_saddr,
+#endif
                               .tos = RT_TOS(tos) } },
                 .proto = IPPROTO_UDP };
            err = ip_route_output_key(&rt, &fl);
@@ -236,7 +240,7 @@
                 goto tx_error;
        }

-#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45)
+#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45) && !defined(LINUX_26)
         mtu = rt->u.dst.pmtu - (cipehdrlen+cipefootlen);
 #else
         mtu = dst_pmtu(&rt->u.dst) - (cipehdrlen+cipefootlen);
@@ -252,7 +256,7 @@
                 tunnel->stat.collisions++;
                 goto tx_error;
        }
-#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45)
+#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45) && !defined(LINUX_26)
        if (skb->dst && mtu < skb->dst->pmtu) {
                 skb->dst->pmtu = mtu;
 #else
@@ -393,7 +397,7 @@
         nf_conntrack_null(skb);
 #if LINUX_VERSION_CODE >= KERNEL_VERSION(2,4,0)
         {
-#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45)
+#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,45) && !defined(LINUX_26)
             int err = NF_HOOK(PF_INET, NF_IP_LOCAL_OUT, skb, NULL,
                               rt->u.dst.dev, ip_send);
 #else
```

Poté je možno jádro přeložit

```
# cd /usr/src
# tar xjf kernel-source-2.4.22.tar.bz2
# export PATCH_THE_KERNEL=NO
```

```
# make-kpkg clean
# make-kpkg --append-to-version -yoda --revision 12 \
            --config menu \
            kernel_image modules_image
```

Při překladu nastaly problémy s moduly `alsa-driver`.

```
/usr/bin/make  -C pnp modules
make[5]: Entering directory ‘/usr/src/modules/alsa-driver/support/pnp’
make[5]: *** No rule to make target ‘/usr/src/linux/include/linux/modules/bm_osl.ver’, needed
make[5]: Leaving directory ‘/usr/src/modules/alsa-driver/support/pnp’
make[4]: *** [_modsubdir_pnp] Error 2
make[4]: Leaving directory ‘/usr/src/modules/alsa-driver/support’
make[3]: *** [compile] Error 1
make[3]: Leaving directory ‘/usr/src/modules/alsa-driver’
make[2]: *** [build-stamp] Error 2
make[2]: Leaving directory ‘/usr/src/modules/alsa-driver’
make[1]: *** [kdist_image] Error 2
make[1]: Leaving directory ‘/usr/src/modules/alsa-driver’
Module /usr/src/modules/alsa-driver failed.
Hit return to Continue
```

6.9.3.3. Ukázka překladu jádra 2.4.24

```
# tar xjf kernel-source-2.4.24.tar.bz2
# ln -s kernel-source-2.4.24 linux
# make-kpkg --append-to-version -yoda --revision 1 \
            --config menu \
            kernel_image
```

6.9.4. Nezapracované texty

6.9.4.1. Překlad jádra 2.4.21 ve Woody bez záplat.

Přejmenujeme adresář se záplatama

```
# mv kernel-patches kernel-patches.off
```

a pak postupujeme standardně.

Mělo by také fungovat nastavení proměnné `PATCH_THE_KERNEL` na `NO` před spuštěním **make-kpkg**.

6.9.4.2. Konfigurace těsně před překladem

Těsně před překladem můžeme jádro konfigorovat použitím volby `--config`

```
# make-kpkg --append-to-version -kertok --revision 1 --config menuconfig \
            kernel_image modules_image
```

6.9.5. Program make-kpkg

* *section id="make-kpkg"*
* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*
* *print="**psselect -p??-?? unix.ps|foldprn -s12**"*

Program se nachází v balíčku `kernel-package` a jeho hlavním účelem je připravovat binární balíčky s jádrem. Tedy řídí celou kompilaci a vytváření balíčku `kernel-image-????`.

Konfigurace je v souboru `/etc/kernel-pkg.conf`. Tento jsem si upravil.

```
maintainer := Radek Hnilica
email := radek@hnilica.cz

patch_the_kernel := YES
```

`--revision` *`číslo_revize`*

Číslo revize může použít znaky `a-z`, `0-9` a znaky „+“, „`.`“ a musí obsahovat alespoň jednu číslici.

`--append_to_version` *`název`*
`--append-to-version` *`název`*

Řetězec bude připojen za číslo verze jako jeho rozšíření. Například `--append-to-version -futok` připojí za číslo verze řetězec „-futok“. Povšimněte si pomlčky která pak odděluje číslo verze od námi specifikovaného rozšíření.

`--flavour` *`název`*

Nepoužívat. Používat místot této volby volbu `--append-to-version`

`--added-modules` *`seznam_modulů`*
`--added_modules` *`seznam_modulů`*

FIXME:

`--added-patches` *`seznam_záplat`*
`--added_patches` *`seznam_záplat`*

Jako argument je seznam čárkou oddělených záplat. Pro správnou funkci vyžaduje nastavení parametru `patch_the_kernel` v souboru `/etc/kernel-pkg.conf` na hodnotu `YES`. Proměnná prostředí `PATCH_THE_KERNEL` má přednost před nastavením v konfiguračním souboru.

`--arch` *`architektura`*

FIXME:

`--cross-compile` *`architektura`*
`--cross_compile` *`architektura`*

FIXME:

`--subarch` *`architektura`*

FIXME:

`--arch-in-name` *`architektura`*
`--arch_in_name` *`architektura`*

FIXME:

`--pgpsign` *`jméno`*

FIXME:

`--config` *`cíl`*

FIXME:

`--targets`

FIXME:

`--noexec`

FIXME:

`--initrd`

FIXME:

`--zimage`

FIXME:

`--bzimage`

FIXME:

`--rootcmd` *`foo`*

FIXME:

`--us`

FIXME:

`--uc`

FIXME:

6.9.6. Script kpatchup

* *section id="fragment.section_template" condition="author"*

kpatchup je skript jenž se stará o záplatování jádra. Dozvěděl jsem so o něm z Kernel Traffic #256

* *This is the first release of kpatchup, a script for managing switching between kernel releases via patches with some smarts:*

- understands -pre and -rc version numbering
- aware of various external trees
- automatically patch between any tree in an x.y release
- automatically download and cache patches on demand

- automatically determine the latest patch in various series
- optionally print version strings or URLs for patches

Currently it knows about 2.4, 2.4-pre, 2.6, 2.6-pre, 2.6-bk, 2.6-mm, and 2.6-tiny. Příklad použití:

```
$ head Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 2
EXTRAVERSION =-rc2
[...]
$ kpatchup 2.6-mm
2.6.2-rc2 -> 2.6.4-rc1-mm1
Applying patch-2.6.2-rc2.bz2 -R
Applying patch-2.6.2.bz2
Applying patch-2.6.3.bz2
Downloading patch-2.6.4-rc1.bz2...
Applying patch-2.6.4-rc1.bz2
Downloading 2.6.4-rc1-mm1.bz2...
Applying 2.6.4-rc1-mm1.bz2
$ head Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 4
EXTRAVERSION =-rc1-mm1
NAME=Feisty Dunnart
[...]
$ kpatchup -q 2.6.3-rc1
$ head Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 3
EXTRAVERSION =-rc1
NAME=Feisty Dunnart
[...]
$ kpatchup -s 2.6-bk
2.6.4-rc1-bk3
$ kpatchup -u 2.4-pre
http://www.kernel.org/pub/linux/kernel/v2.4/testing/patch-2.4.26-pre1.bz2
This is an alpha release for people to experiment with. Feedback and
patches encouraged. Grab your copy today at:

    http://selenic.com/kpatchup/

Zwane Mwaikambo was very excited by this, saying, "Oh i definitely owe you one
now, this is replacing the ugly shell script i had before, i'm mostly using
this now to download and patch up trees before cvs import'ing them." Rusty
Russell was also happy to see this, and offered his own scripts in case they
had anything worth merging. Dave Hansen also liked Matt's script, but said:

    it doesn't look like it properly handles empty directories. I tried this
    command, this morning, and it blew up. I think it's because this directory
    http://www.kernel.org/pub/linux/kernel/v2.6/snapshots/ is empty because of
    last night's 2.6.4-rc2 release. I don't grok python very well but is the
    "return p[-1]" there just to cause a fault like this? Would it be better if
    it just returned a "no version of that patch right now" message and exited
    nicely?
```

```
[dave@nighthawk linux-2.6]$ kpatchup-0.02 2.6-bk
"Traceback (most recent call last):
  File "/home/dave/bin/kpatchup-0.02", line 283, in ?
    b = find_ver(args[0])
  File "/home/dave/bin/kpatchup-0.02", line 240, in find_ver
    return v[0](os.path.dirname(v[1]), v[2])
  File "/home/dave/bin/kpatchup-0.02", line 147, in latest_dir
    return p[-1]
IndexError: list index out of range

I think your script, combined with Rusty's latest-kernel-version could make
me a very happy person.

They debugged for a bit, and the thread ended.
```

6.9.7. Ukázky překladu jádra linuxu

* *section id="ukazky-prekladu-jadra-linuxu"*
* *print="**psselect -p 72-75 unix.ps|foldprn -s4**"*

Zdroje a odkazy:

- **FIXME:**

ToDo list

1. **FIXME:**

Zde uvádím několik konkrétních postupů překladu jádra.

6.9.7.1. Jádro 2.4.25 pro počítač yoda

Na tomto stroji provádím překlad pomocí jednoduchého skriptu `/root/sbin/build-kernel`:

```
#!/bin/sh
export PATCH_THE_KERNEL=NO
cd /usr/src
rm -fr modules
tar xjf alsa-driver.tar.bz2
tar xzf cipe.tar.gz
tar xzf thinkpad.tar.gz
tar xzf pcmcia-cs.tar.gz
cd linux
make-kpkg clean
make-kpkg --append-to-version -$1 --revision $2 --config menu kernel_image modules_image
```

Skript připraví k překladu i moduly pro tento počítač specifické.

Zdroje jádra musí být připraveny včetně symbolického odkazu `linux`

```
# cd /usr/src
# tar xjf kernel-source-2.4.25.tar.bz2
# rm linux
# ln -s kernel-source-2.4.25 linux
# cd linux
# cp /boot/config-2.4.24-yoda yoda.0
```

```
# /root/sbin/build-kernel yoda 3
```

V konfiguraci jsem načetl konfiguraci aktuálního jádra z `/boot/config-2.4.24-yoda` a po několika změnách (aktivoval jsem ACPI) jsem ji uložil jako `yoda.1`. Překlad proběhl úspěšne a vytvořili se balíčky:

```
# ls -l ../*.deb
-rw-r--r--    1 root     src        211710 Mar 17 11:57 ../alsa-modules-2.4.25-yoda_0.9.8-3+3_
-rw-r--r--    1 root     src         33380 Mar 17 11:54 ../cipe-2.4.25-yoda_1.5.4free-7+3_i386
-rw-r--r--    1 root     src       2878496 Mar 17 11:54 ../kernel-image-2.4.25-yoda_3_i386.deb
-rw-r--r--    1 root     src        356126 Mar 17 11:55 ../pcmcia-modules-2.4.25-yoda_3.1.33-6
-rw-r--r--    1 root     src         18828 Mar 17 11:57 ../thinkpad-modules-2.4.25-yoda_4.9-1+
```

Přistoupil jsem tedy k instalaci

```
# cp ../*.deb /root/debs/
# # update-debs
 ** Packages in archive but missing from override file: **
  alsa-modules-2.4.24-yoda alsa-modules-2.4.25-yoda bluez-hcidump
  cipe-2.4.24-yoda cipe-2.4.25-yoda irate-client-gtk irate-client-
  motif kernel-image-2.4.24-yoda kernel-image-2.4.25-yoda
  libbluetooth1 libbluetooth1-dev pcmcia-cs pcmcia-source springgraph
  thinkpad-modules-2.4.24-yoda thinkpad-modules-2.4.25-yoda

 Wrote 16 entries to output Packages file.
# apt-get update
# apt-get install kernel-image-2.4.25-yoda
```

Informoval jsem o novém jádru zavaděč

```
# vi /boot/grub/menu.lst
```

A v očekávání restartoval počítač.

6.9.7.2. Nasazení jádra `2.6.0-test9`

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Nejdříve si rozbalíme zdroje.

```
# cd /usr/src
# tar xjf kernel-source-2.6.0-test9.tar.bz2
# ln -s kernel-source-2.6.0-test9 linux-2.6.0
# ln -s linux-2.6.0 linux
```

Pak linux nakonfigurujeme.

```
# cd linux
# make menuconfig
```

A poté zkompilujeme jádro a moduly.

```
# export PATCH_THE_KERNEL=NO
# make-kpkg clean
# make-kpkg --append-to-version -yoda --revision 1 kernel_image modules_image

# export PATCH_THE_KERNEL=NO
# make-kpkg clean
# make-kpkg --append-to-version -yoda --revision 1 kernel_image
```

6.9.7.3. Nasazení jádra 2.6.x

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

6.9.7.3.1. Použití Debian Backports (Backports.ORG) (http://www.backports.org/)

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Do `/etc/apt/sources.list` si přidáme potřebné zdroje

```
deb file:/home/mirror/backports/debian woody kernel-source-2.6.0 kernel-package
```

a nainstalujeme zdroje a potřebné nástroje.

```
# apt-get update
# apt-get install kernel-source-2.6.0 kernel-tree-2.6.0 kernel-doc-2.6.0 kernel-patch-2.6.0
# apt-get install --reinstall kernel-package
```

V dokumentaci k balíčku `kernel-package` v adresáři `/usr/share/doc/kernel-package` jsou pak popsány postupy.

6.9.7.3.1.1. Překlad pod uživatelem

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
$ mkdir ~/kernel
$ cd ~/kernel
$ tar xjf /usr/src/kernel-source-2.6.0.tar.bz2
$ cd kernel-source-2.6.0
$ make menuconfig
$ make-kpkg clean
$ fakeroot make-kpkg --revision=yoda.1 kernel_image
```

Přeložení modulů

```
$ export MODULE_LOC=$HOME/kernel/modules
$ FIXME
```

6.9.7.4. Překlad jádra 2.4.26 s podporou HTB

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Jádro překládám na virtuánlím stroji deb z balíčků z backports.org (http://www.backports.org). Do `/etc/apt/sources.list` přidám:

```
deb http://localhost:9999/backports woody kernel-source-2.4.26 kernel-package
```

a nainstaluji

```
# apt-get install kernel-source-2.4.26 kernel-package libncurses-dev
```

a zkusím překlad

```
# cd /usr/src
# tar xjf kernel-source-2.4.26.tar.bz2
# ln -s kernel-source-2.4.26 linux
# export PATCH_THE_KERNEL=NO
# cd linux
# make-kpkg clean
# make-kpkg --append-to-version -mamlas --revision 4 --config menu \
            kernel_image modules_image
```

Potřebné iproute se pokusím nainstalovat z backports.org (http://www.backports.org) ještě předtím než je budu případně kompilovat. Do zdrojů `/etc/apt/sources.list` si přidám řádek

```
deb http://www.backports.org/debian/ woody iproute
```

a nainstaluju

```
# apt-get update
# apt-get install iproute
```

6.9.7.5. Překlad jádra 2.4 + vserver z Debian Backports (Backports.ORG) (http://www.backports.org/)

* *$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $*

V instlačních zdrojích v souboru `/etc/apt/sources.list` musíme mít řádky:

```
deb http://localhost:9999/backports woody kernel-2.4
deb http://debian:9999/main sarge main
```

První nám zpřístupní zdroje jádra 2.4.27 z Debian Backports (Backports.ORG) (http://www.backports.org/) a druhý nám umožní nainstalovat `kernel-patch-ctx` potřebný pro VServer.

```
deb:~# apt-get install -t backports kernel-source-2.4.27
deb:~# apt-get install dh-kpatches
deb:~# apt-get install kernel-patch-ctx
```

Nyní můžeme přeložit jádro:

```
deb:~# cd /usr/src
deb:~# tar xzf kernel-source-2.4.27_2.4.27.orig.tar.gz
deb:~# ln -s kernel-source-2.4.27-2.4.27 linux
deb:~# cd linux
deb:~# export PATCH_THE_KERNEL=YES
deb:~# make-kpkg clean
deb:~# cp kernel-config-file-from-some-where .
deb:~# make-kpkg --append-to-version -sunset-vs --revision 1 \
          --added-patches vserver \
          --initrd \
          --config menu kernel_image modules_image
```

* *Dále následují nezpracované poznámky.*

deb http://localhost:9999/backports woody kernel-package modutils
#deb http://localhost:9999/backports woody kernel-source-2.4.26
#deb http://localhost:9999/backports woody kernel-source-2.4.27

util-vserver 0.30-14
vserver-debiantools 0.1.9, 0.1.10

6.9.7.6. Jádro 2.4.27 z Debian Backports (Backports.ORG) (http://www.backports.org/) a lm-sensors

```
# apt-get update
# apt-get install kernel-source-2.4.27 kernel-package kernel-patch-2.4-i2c \
                  kernel-patch-2.4-lm-sensors dh-kpatches
Reading Package Lists... Done
Building Dependency Tree... Done
Sorry, kernel-patch-2.4-i2c is already the newest version.
Sorry, kernel-patch-2.4-lm-sensors is already the newest version.
Sorry, kernel-package is already the newest version.
The following extra packages will be installed:
  libsensors3 ucf
The following NEW packages will be installed:
  kernel-source-2.4.27 libsensors3 lm-sensors ucf
```

Configuration of lm-sensors

lm-sensors requires kernel support to access sensor devices.

To know how to configure your kernel for your motherboard, please look /usr/share/doc/lm-sensors/README.Debian.

```
gembird:/usr/src# tar xjf kernel-source-2.4.27.tar.bz2
# ln -s kernel-source-2.4.27 linux
# cd linux
# export PATCH_THE_KERNEL
# make-kpkg clean
# cp /boot/config-2.2.20-idepci .  # Konfigurace aktuálního jádra
# make-kpkg --append-to-version -gembird --revision 1 \
            --added-patches i2c lm-sensors \
            --initrd --config menu kernel_image modules_image
```

6.9.7.7. Překlad jádra 2.4.27 z Debian Backports (Backports.ORG) (http://www.backports.org/) z řadou patchů

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Poznámky k překladu jádra již narostly a začaly být nepřehledné. Tímto se pokouším je trochu uspořádat. Začal jsem při potřebě přeložit jádro pro server. Zkusím si vytvořit jádro které budu moci nasadit na všechny servery a tím si zjednodušit administraci těchto serverů.

Překlad je prováděn na čistém virtuálním serveru `bone` instalovaném z Debina Woody. Použití tohoto virtuálního serveru má zajistit že se mi do překladu nezamíchají věci ze Sarge. Rovněž mi umožní sledovat které balíčky jsou opravdu nezbytné pro správný překlad.

Na čistém serveru jsem opravil soubor `/etc/apt/sources.list`. Mimo základní zdroje Woody jsem přidal zdroje z Debian Backports (Backports.ORG) (http://www.backports.org/) nutné pro překlad novějšího jádra a taky s novějšími patchi. Jako zdrojový server používám apt-proxy bežící na stejném stroji.

```
deb http://localhost:9999/main woody main contrib non-free
deb http://localhost:9999/non-US woody/non-US main contrib non-free
deb http://localhost:9999/security woody/updates main non-free
deb http://localhost:9999/main sarge main       # kernel-patch-vserver
deb http://localhost:9999/backports woody kernel-2.4
...

# apt-get update
# apt-get upgrade
# apt-get install gcc libncurses5-dev
# apt-get install kernel-source-2.4.27 kernel-package dh-kpatches
# apt-get install kernel-patch-vserver
```

Po nainstalování pak rozbalíme zrdroje a spustíme překlad.

```
# cd /usr/src
# tar xjf kernel-source-2.4.27.tar.bz2
# ln -s kernel-source-2.4.27 linux
# cd linux
# export PATCH_THE_KERNEL=YES
# make-kpkg clean
# cp kernel-config-file-from-some-where .
# make-kpkg --append-to-version -vs --revision 1 \
            --added-patches vserver \
            --initrd --config menu kernel_image modules_image
```

V konfiguračním menu jsem nejdříve načetl volby z nakopírovaného onfiguračního souboru. To má zajistit aby nově sestavované jádro mělo všechny původní vlastnosti a ovladače. Poté jsem provedl jemné vyladění parametrů jádra. Mimo jiné jsem nastavil parametr `CONFIG_BLK_DEV_VROOT` na „m“. Tento parametr najdeme v menu Block devices⟶Virtual Root device support (NEW)

Ještě uvedu skript jenž jsem užíval k překladu.

```
#!/bin/sh
# $Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $
# Build kernel for Debian Woody
# Copyright (c) 2005 Radek Hnilica

VARIANT=tvs
REVISION=${1:-1}

cd /usr/src
rm -fr kernel-source-2.4.27 linux

tar xjf kernel-source-2.4.27.tar.bz2
ln -s kernel-source-2.4.27 linux
cp $VARIANT.* linux

cd linux
export PATCH_THE_KERNEL=YES
```

```
cp ../config-2.4.28-bf2.4 dell
make-kpkg clean
make-kpkg --append-to-version -$VARIANT --revision $REVISION \
          --added-patches vserver \
          --initrd --config menu kernel_image modules_image
```

6.9.7.8. Překlad jádra 2.6.11.9 na SuSE

FIXME:

```
# tar xjf linux-2.6.11.9.tar.bz2
# ln -s linux-2.6.11.9 linux
# cp /boot/config-2.6.5-7.97-default linux/.config
# cd linux
```

Případné aplikování záplat:

```
# patch -p1 < soubor_se_záplatou.patch
```

A můžeme začít:

```
# cd /usr/src/linux
# make menuconfig  # config/menuconfig/xconfig
```

V konfiguraci nastvíme co potřebujeme.

Loadable module support / Enable loadable module support
⋮

```
# make clean
# make bzImage
# make modules
```

Přeložené jádro nainstalujeme:

```
# make modules_install
# cp arch/i386/boot/bzImage /boot
# mkinitrd -k /boot/bzImage -i /boot/initrd.test
Root device:    /dev/hda3 (mounted on / as ext3)
Module list:    jbd ext3

Kernel image:   /boot/bzImage
Initrd image:   /boot/initrd.test
Shared libs:    lib/ld-2.3.3.so lib/libc.so.6 lib/libselinux.so.1
Cannot determine dependencies of module jbd. Is modules.dep up to date?
Cannot determine dependencies of module ext3. Is modules.dep up ro date?
Modules:
none
# depmod -a
# #
# mkinitrd -k /usr/src/linux/arch/i386/boot/bzImage -i /tmp/initrd
```

Budeme potřebovat: gcc, make, m4, ncurses-devel, patch, . . .

Poznámky:

```
# uname -r
```

```
# uname -a
```

6.9.8. Init RAM disk

FIXME:

6.9.9. České prostředí

Konfigurace českého prostředí sestává z konfigurace řady subsystémů. Jedná so o vstup (česká klávesnice na konsole a v X), výstup (české fonty na konsole a v X) a prostředí (programy vypisují hlášení česky a nikoliv anglicky).

- vstup —
- výstup —
- prostředí —

Globální nastavení českého prostředí a nastavení pro uživatele.

Do souboru `/etc/environment` můžeme vložíme řádku

```
LANG=cs_CZ
```

ale lépe to nechat na systém. České prostředí nastavíme pomocí

```
# dpkg-reconfigure locales
```

kde si vybereme české prostředí `cs_CZ.ISO-8859-2`

Do globálního profilu, soubor `/etc/profile` vložíme

```
set meta-flag on
set convert-meta off
set output-meta on
```

6.9.9.1. Česká (národní) klávesnice na konsole

* *Podle příspěvku „ceska klavesnice v konzoli“ zaslaného do news://cz.comp.linux.debian Karlem Benesem*

Klávesnici nastavíme příkazem

```
# dpkg-reconfigure console-common
```

* *Mapa klávesnice ja na http://www.matfyz.cz/hands/Unix/Console/Keyboard.*

* *Podle příspěvku „Re: Ceska klavesnice jako implicitni?“ zaslaného do `cz.comp.linux.debian` Hansem Ginzelem .*

Přímé *naloadování* klávesové mapy:

```
# loadkeys /usr/share/keymaps/i386/qwerty/cz-lat2-prog.map.gz
```

Při hledání chyby je vhodné vědět jaký kód (scancode, keycode) které tlačítko vysílá:

```
# showkey -s
# showkey -k
```

Nezmáčkneme-li po dobu asi 10sec žádnou klávesu, program se ukončí.

Aktuální mapování kláves zjistíme

```
# dumpkeys -l
```

Sprovoznění zvláštních kláves (WakeUp, PowerDown, Sleep). Do `/etc/init.d/keymap.sh` přidat na začátek řádek

```
. $CONFDIR/$SCANCODES
```

kde

```
$SCANCODES=add_scancodes
```

Do souboru `/etc/console/add_scancodes` napíšeme

```
#           WakeUp    Sleep     Powerown
setkeycodes e063 111  e05f 107  e05e 109
```

* *Podle příspěvku „Re: Ceska klavesnice jako omplicitni?“ zaslaného do news://cz.comp.linux.debian Hasem Ginzelem*

S klávesovými mapami se dá dále kouzlit. Například uvedením

```
string F105 = "\033[cl"
control keycode 105 = F105  # Ctrl-Left
```

se zajistí, že Control+Šipka doleva vyšle uvedenou escape sekvenci.

Klávesové mapy se nalézají v `/usr/share/keymaps/`

Poznámka: Nastavení české klávesnice na konsoli v Debian Etch od Vít Baloun:

```
# aptitude install kbd locales
# dpkg-reconfigure locales
```

(a zvoleno cs_CZ.UTF-8 jako default a en_US.UTF-8 jako alternativa)

6.9.9.2. Česká (národní) fonty na konsole

* *Podle příspěvku „Re: Opet cestina“ zaslaného do news://cz.comp.linux.debian Hasem Ginzelem*

```
# consolechars -f iso02.f16
# charset G0 iso02+euro
```

Je možno použít ucw fonty od Martina Mareše ftp://atrey.karlin.mff.cuni.cz/pub/local/mj/linux/ucw-cs-1.1.tar.gz. Toho docílíme nastavením v souboru `/etc/console-tools/config`, kde uvedeme

```
SCREEN_FONT=ucw16
APP_CHARSET_MAP=iso02+euro
```

Změny promítneme do běžícího systému voláním

```
# /etc/init.d/console-screen.sh restart
```

Fonty a mapování se nalézají v `/usr/share/console*/`

6.9.9.3. Česká (národní) klávesnice v X

FIXME: doplnit

* *Hans Ginzel*

Část (relevantní) konfiguračního souboru `/etc/X11/XF86Config-4`

```
Section "InputDevice"
        Identifier      "Configure Keyboard"
        Driver          "keyboard"
        Option          "CoreKeyboard"  "true"
        Option          "XkbRules"      "xfree86"
        Option          "XkbModel"      "pc102"
        #Option         "XkbLayout"     "czsk(us_cz_prog)"
        Option          "XkbLayout"     "cs_CZ"
        Option          "XkbVariant"    "us"
        Option          "XkbCompat"     "group_led"
EndSection
```

6.9.9.4. Česká (národní) fonty v X

FIXME: doplnit

* *Hans Ginzel*

Změníme aliasy v souboru `/etc/X11/fonts/misc/xfonts-base-il2.alias`

```
fixed           -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-2
variable        -*-helvetica-bold-r-normal-*-*-120-*-*-*-*-iso8859-2
5x7             -misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-2
5x8             -misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-2
```

Poté spustíme

```
# update-fonts-alias misc
```

* *Petr Vaněk*

Instalované balíčky s fontama

```
gsfonts
gsfonts-x11
```

```
xfonts-100dpi
xfonts-100dpi-transcoded
xfonts-75dpi
xfonts-75dpi-transcoded
xfonts-abi
xfonts-base
xfonts-base-transcoded
xfonts-biznet-100dpi
xfonts-biznet-75dpi
xfonts-biznet-base
xfonts-biznet-iso-8859-2-100dpi
xfonts-biznet-iso-8859-2-75dpi
xfonts-biznet-iso-8859-2-base
xfonts-intl-european
xfonts-scalable
```

Chceme li použít TrueType fonty od Microsotu, nainstalujeme `msttfcorefonts`

Obsah sekce FontPath v konfiguračním souboru X

```
FontPath    "/usr/X11R6/lib/X11/fonts/misc/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
```

6.9.9.5. Česká (národní) prostředí

FIXME: doplnit

„Národnost“ prostředí je určena nastavením locales. Nejdůležitější je nastavení v souboru `/etc/environment` které je implicitní pro celý systém. Soubor needirujeme přímo ale nastaveníprovádíme příkazem

```
# dpkg-reconfigure locales
```

České nastavení se pak projeví řádkem

```
LANG=cs_CZ
```

v tomto souboru.

6.9.9.6. Nastavení národního (českého) prostředí v některých programech

6.9.9.6.1. bash

Chceme-li používat v shellu české či jiné národní znaky musíme povolit osmibitové znaky, jejich vstup a zobrazení. Toho dosáhneme přidáním následujících řádků do souboru `/etc/inputrc`

```
set input-meta on
set output-meta on
set convert-meta off
```

* *see* ***info rluserman***

6.9.10. Postupy

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Downgrade

```
# apt-get clean
# apt-get --reinstall install seznam_balíčků
```

6.9.10.1. Downgrade z Testing na Stable

* *Na DebianPlanet (http://www.debianplanet.org) byl zveřejněn článek How I Downgraded Testing to Stable (http://www.debianplanet.org/node.php?id=880) ze kterého vycházím.*

Nejdříve opravíme `/etc/apt/sources.list`, musí obsahovat odkazy na stabilní balíčky

```
### Stable -- Woody
deb http://debian:9999/main woody main contrib non-free
deb http://debian:9999/non-US woody/non-US main contrib non-free
deb-src http://debian:9999/main woody main contrib non-free
deb-src http://debian:9999/non-US woody/non-US main contrib non-free
deb http://security.debian.org/ woody/updates main contrib non-free
```

a `/etc/apt/apt.conf.d/70debconf` kde nastavíme defaultní distribuci na `stable`

```
APT::Default-Release "stable";
```

Tímto máme zajištěno že se již nebudou instalova balíčky z `testing` a otevřeme si cestu pro instalaci ze `stable`.

Jeden ze čtenářů doporučuje downgrade takto

```
# for x in ‘dpkg --get-selections | awk ’{print $1}’‘; do
>     cat /var/lib/apt/lists/debianmirror_*_stable_Packages | grep ^Package: $x$
> done | awk ’{ print $1 "/stable" }’ >stablestuff

# apt-get install ‘cat stablestuff‘
```

Další pak doporučuje jednodušší postup

```
# apt-get update
# apt-get --reinstall -y install ‘dpkg --get-selections‘
```

jiný opravil tento postup. Místo `‘dpkg -get-sellections‘` dát

```
dpkg --get-selections | awk -F {print $1}
```

nebo

```
dpkg --get-selections | sed -e ’s/deinstall//’ | sed -e ’s/install//’

apt-get --reinstall -y install ‘dpkg --get-selections|grep -v deinstall|cut -f 1‘
```

6.9.10.2. Duplikování hotové instalace

Rozmnožování

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Pokud potřebujeme nainstalovat více stejných strojů, můžeme použít následující postup:

1. Nainstalujeme a nakonfigurujeme jeden vzorový stroj
2. Mají-li další stroje stejné disky, použijeme přímé kopírování disků. Tedy cílový disk připojíme jako další do počítače a zadáme příkaz

   ```
   # dd if=/dev/hda of=/dev/hdc
   ```

   kde `/dev/hda` je disk s nainstalovaným a nakonfigurovaným systémem a `/dev/hdc` je disk cílový.
3. Nejsou-li disky stejné, přípojíme nový disk do počítače, rozdělíme a vytvoříme oddíly. Cílový oddíl namountujeme jako `/target` a zadáme

   ```
   # cp -ax / /target
   ```

   poté smažeme všechny adresáře jenž mají zůstat prázdné, jako například `/proc`

   ```
   # rm -r /target/proc/*
   ```

V konferenci *debian-isp* se objevila tato zpráva o množení instalace.

> For an ADSL router that we like to copy:
>
> The working system is on hda
>
> we put a 250MB or more disk as hdc
>
> and run a script to
> copy hda to hdc
> lilo hdc so that it will boot as hda
> make some changes in /etc
>
> Much as I like Debian, dselect and all that,
> sometimes there is just a simpler way to do stuff.
>
> Regards
> ragnar@this.is

6.9.10.3. Instalace na více počítačů

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Pro instalaci na více počítačů můžeme použít program replicator. Nainstlujeme jej

```
# app-get install replicator
```

Další informace jsou na Source Forge (http://replicator.sourceforge.net/).

Další možností je program fai který je v balíčku fai-kernels nebo také na Source Forge (http://fai.sourceforge.net/).

* *Prozkoumat a popsat použití*

6.9.10.3.1. Masová instalace

Mass installation procedure for Debian

1. Nabootujte z disket
2. Nahrajte `base.tgz` přes síť z instalačního serveru. (*mirror server*)
3. Follow prompts on debian setup to setup network, DNS, apt sources, root password, user account and password etc.
4. Nahrajte `tar.gz` kde jsou *customized* věci. Rozbalte do `/etc`, `/usr/local` a `/var/www`.
5. spusťte

   ```
   # dpkk --set-sellections </etc/deblist
   ```

   kde soubor `/etc/deblist` je z naší *tarball*
6. Spusťte **apt-get** a nechte nainstalovat potřebné balíčky. Všimněte si, že **dpkg** se ptá jestli přepsat konfigurační soubory v `/etc`. Odpovězte NE (*NO*)
7. Upravte potřebné konfigurační soubory v `/etc`. Například jestli je naše tarová koule (*tarball*) pro `customerdomain.com`, vyhledáme a zaměníme všechny výskyty za skutečnou/opravdovou doménu. Můžete použít **mc** a ručně zkontrolovat každou výměnu/náhradu, jen tak pro jistotu.
8. Jestli potřebujete balíčky, které nejsou standardně nainstalovány, nyní je nainstalujte. Toto se týka často také *customized* jádra.
9. Každý stroj/služba je plně testována. DNS, dhcp, samba, isp dial-out, ras dial-in, mail in, mail out, proxy server, atd.
10. Detaily nastavení stroje zdokumentujeme, a tento je připraven pro odeslání.

6.9.11. System V init skripty

FIXME:dopsat popis init skriptů.

Adresáře `/etc/init.d/`

`/etc/init.d`

FIXME:

`/etc/rcN.d`

FIXME:

`/etc/init.d/rcN.d`
`/etc/rc.d/rcN.d`

FIXME:

`/etc/init.d`

FIXME:

Runlevely

Popis runlevelů

0

FIXME: Halt machine

1

FIXME: single user, maintenance

2

FIXME: multiuser, default for Debian

3

FIXME: multiuser, default for RH, SUSE, ...

4

FIXME: nepoužívá se

5

FIXME: xdm

6

FIXME: Reboot machine

S

FIXME:

Změnu runlevelu, tedy přechod z jednoho do jiného zajistíme programem **telinit** (**init**). Spusíme jej s parametrem označujícím cílový runlevel.

```
# telinit 1
```

Parametrem je buď to číslo levelu (0-6) nebo „s“ jako alias pro runlevel 0. Speciálním parametrem je „q“ nebo „Q“ kterým se **init** procesu oznámí že byla upravena tabulka `/etc/inittab` a že ji má znovu načíst.

6.9.12. Ruční vytvoření základní instalace

Odkazy a zdroje:

- balíček `debootstrap`
- http://www.obsidian.com.au/vserver/mini-debian-vservers-016.txt

FIXME:

6.9.13. Downgrade

Někdy se nám stane, že potřebujeme downgradovat balíček z vyšší verze na verzi nižžší. Například po upgradu ze stable na testing zpátky. Mě se jednou podařilo chybou v konfiguraci upgradovat libc6 ze zmíněného `stable` na `testing`. Downgrade jsem provedl po opravě v konfiguračních souborech takto:

```
# apt-get install libc6/stable libc6-dev/stable locales/stable
```

Downgrade těchto balíčků z `sarge` na `woody` byl bezproblémový.

6.9.14. Zdroje balíčků

V krátkosti si povíme odkud je možno instalovat.

Hlavními instalačními zdroji jsou archívi debianu a jejich zrcadla rozmístněná po celém světě.

- http://www.debian.org

6.9.14.1. Backports

Projekt Backports (http://www.backports.org) vznikl někdy v roce 2003 a jeho. Jeho cílem je bortovat do stabilní verze programy novější z verze testovací a verze nestabilní. Dává nám tedy možnos s přijetím rizika používat novější software, aniž bychmom museli svůj počítač upgradovat na testing či stable, nebo překládat si požadovaný software sami.

Vybrané zdroje balíčků z backports samozřejmě můžeme používat i s využitím apt-proxy. V konfiguračním souboru `apt-proxy.conf` uvedeme:

```
add_backend /backports/                                 \
$APT_PROXY_CACHE/backports/                             \
http://www.backports.org/debian/



>From my sources.list:
,----
| deb http://10.0.0.1:9999/backports woody kernel-2.6
`----

>From `apt-cache policy`:
,----
|  900 http://10.0.0.1 stable/main Packages
|      release v=3.0r2,o=Debian,a=stable,l=Debian,c=main
|      origin 10.0.0.1
|
|  500 http://10.0.0.1 woody/kernel-2.6 Packages
|      origin 10.0.0.1
`----
```

6.9.15. Správa balíčků

Debian GNU/Linux používá systém balíčků DEB.

6.9.15.1. Vyhledání

```
$ zgrep co_hledám .../Contents-i386.gz

$ grep co_hledám /var/lib/dpkg/available
```

```
$ apt-cache search libgnutils

$ dpkg -S /bin/sh

$ dlocate

$ dpkg -L název_balíčku
```

6.9.16. apt

* ***Attributy:*** *id="apt"*
* *Klony apt-proxy. Prozkoumat, popsat aptcached (http://talk.trekweb.com/~jasonb/software.shtml), `apt-proxy2` (http://lug-owl.de/~jbglaw/software/ap2), `apt-www-proxy` (http://ironsides.terrabox.com/~ahzz/apt-www-proxy/index.html) a `debproxy` (http://sourceforge.net/projects/debproxy).*
* *`# apt-get --reinstall install`*

Pár příkladů řádků do souboru `/etc/apt/sources.list`. Nejdříve originální řádky

```
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org/ stable/updates main contrib non-free
```

Vše co se týká konfigurace ohledně apt se nachází v adresáři `/etc/apt`. Jedná se hlavně o soubor s odkazy na standardní repositáře, který vznikne po instalaci `/etc/apt/sources.list`, soubro s preferencemi `/etc/apt/preferences`, adresář s konfigurací `/etc/apt/apt.conf.d`, adresář s ostatními repositáři `/etc/apt/sources.list.d` a soubory s gpg klíči.

> **Poznámka:** Adresář `/etc/apt/sources.list.d` se objevil v Debian Lenny a je to preferované místo kam zapisovat další repositáře. Tedy už je nepíšeme přímo do `/etc/apt/sources.list`, ale každá repositář má vlastní soubor v `/etc/apt/sources.list.d`.

6.9.16.1. Instalování balíčků z unstable do testing

Do souboru `/etc/apt/apt.conf` přidáme řádek

```
APT::Default-Release "testing";
```

a balíček z unstable pak i se závislostma z unstable nainstalujeme takto

```
# apt-get -t unstable install balíček
```

Samozřejmě že je třeba mít v `/etc/apt/sources.list` nastaveny zdroje k testing, například takto

```
##### Binaries for Unstable
deb ftp://debian_server/debian unstable main contrib non-free
deb ftp://debian_server/debian-non-US unstable/non-US main contrib non-free
```

Odkazy:

- Jednoduché míchání unstable a testing (http://www.penguin.cz/novinky-view.php3?id=458)

- Mixing Debian releases the easy way (http://www.debian.org/News/weekly/2002/4/mail)

Ve Woody je to rochu jinak, soubor `/etc/apt/apt.conf` neexistuje. Řádek pro něj určený umístníme do souboru `/etc/apt/apt.conf.d/70debconf`

```
APT::Default-Release "stable";
```

Zdroje do souboru `/etc/apt/sources.list` můžou směřovat i na lokální cache

```
deb http://debian:9999/main testing main contrib non-free
deb http://debian:9999/non-US testing/non-US main contrib non-free
```

6.9.16.2. apt-proxy

* *section id="apt-proxy" xreflabel="apt-proxy"*

ToDo list

1. Ověřit postup nainstalováním na nějaký server.
2. Rozepsat význam proměnných v konfiguračním souboru.

Instalujeme-li, či udržujeme balíčky na více počítačích v jedné síti, hodí se nám apt-proxy. Funguje podobně jako například webová proxy squid, s tím že je určena pro debianovské balíčky. Snížíme si tedy nároky na přenosovou kapacitu linky. V době upgradu se každý balíček pro všechny stroje stahuje jen jednou, a po určenou dobu zůstává uložen na proxy odkud si jej jednotlivé stroje stahují.

Na vybraném počítači, jenž nám bude sloužit jako proxy cache, nainstalujeme apt-proxy

```
# apt-get install apt-proxy
```

program se spouští přes internetového superdémona inetd a tomu do konfigurace `/etc/inetd.conf` doplíme řádku pro naši proxy. Vybral jsem pro ni port 9999:

```
9999    stream  tcp     nowait.400      aptproxy /usr/sbin/tcpd \
        /usr/sbin/apt-proxy -l /var/log/apt-proxy.log
```

Nyní nám zbývá upravit konfiguraci v souboru `/etc/apt-proxy/apt-proxy.conf`. Vybereme si svazek na kterém máme potřebný prostor který proxy obětujeme. Velikost potřebného prostoru je závislá na konkrétní konfigurace a použití. Na mém serveru ve firmě při níže uvedeném nastavení zabírá prostor cache kolem 6GB.

```
# Change this path if you do not want to keep your cache under var
APT_PROXY_CACHE=/bd/1/apt-proxy
```

Následuje popis zdrojů balíčků. Budeme poskytovat balíčy pro hlavní větev *main*, *non-US*, bezpěčnostní záplaty *security* a balíčky z backports.org *backports*. deb http://www.debian.cz/debian woody main contrib non-free

```
add_backend /main/                                      \
        $APT_PROXY_CACHE/debian/                        \
        ftp.cz.debian.org::debian/                      \
        ftp.de.debian.org::debian/                      \
        ftp2.de.debian.org::debian/                     \
        ftp.uk.debian.org::debian/                      \
        ftp.us.debian.org::debian/                      \
        ftp.debian.org::debian/

add_backend /non-US/                                    \
        $APT_PROXY_CACHE/non-US/                        \
```

```
            ftp.cz.debian.org::debian-non-US/               \
            ftp.de.debian.org::debian-non-US/               \
            ftp.dk.debian.org::debian-non-US/               \
            ftp2.de.debian.org::debian-non-US/              \
            ftp.uk.debian.org::debian/non-US/

    add_backend /security/                                  \
            $APT_PROXY_CACHE/security/                      \
            security.debian.org::debian-security/           \
            non-us.debian.org::debian-security/

    add_backend /backports/                                 \
            $APT_PROXY_CACHE/backports/                     \
            http://www.backports.org/debian/
```

* *Bylo by dobré popsat význam jednotlivých proměnných a to i těch které jsem ve své konfiguraci nepoužil.*

```
CLEANUP_DAYS=128
CLEAN_SWEEP=360
MAX_VERSIONS=9
BACKEND_FREQ=240
RSYNC_TIMEOUT=30
WGET_TIMEOUT=30
RSYNC=rsync
KEEP_STATS=2
```

Na klientech které budou naši proxy používat k instalaci ji uvedeme v instalčních zdrojích `/etc/apt/sources.list`

```
# Apt-proxy cache
deb http://debian:9999/main woody main contrib
deb http://debian:9999/non-US woody/non-US main contrib
deb http://debian:9999/security stable/updates main
```

Ve Woody je verze 1.3, v Sarge je již novější verze 1.9. Je třeba pročíst dokument `/usr/share/doc/apt-proxy/UPGRADING` a opravit konfigurační soubor `/etc/apt-proxy/apt-proxy-v2.conf`.

6.9.16.3. apt-cacher

*

Odkazy:

- caching proxy for Debian packages (http://pwet.fr/man/linux/commandes/apt_cacher)
- How to: Install a Debian/Ubuntu package (.deb) cache server - apt-cacher (http://www.go2linux.org/debian-ubuntu-package-proxy-server)
- Deb Package Cache (http://wiki.aims.ac.za/mediawiki/index.php/Deb_Package_Cache)

6.9.16.4. Instalace balíčků z vlastního adresáře

Máme balíčky, například vlastnoručně skompilované a potřebujeme je předhodit apt. Balíčky máme například v adresáři `/root/debs/wxwin2.3.3py2.1/`.

1. Vytvoříme soubor `Packages` například takto

```
kvark:~/debs# dpkg-scanpackages wxwin2.3.3py2.1 /dev/null \
                                    >wxwin2.3.3py2.1/Packages
```

2. Do souboru `/etc/apt/source.list` přidáme řádky

   ```
   # Lokalni archiv
   deb file:/root/debs wxwin2.3.3py2.1/
   ```

   A to je vše.

Další ukázka, použitá při instalaci kompilovaného jádra.

1. Vytvoříme si adresář pro balíčky. Vycházím z předpokladu že jeden adresář bez členění na podadresáře stačí.

   ```
   moon:~# mkdir debs
   ```

2. Do adresáře si nakopírujeme zkompilovné balíčky.

   ```
   moon:~# cd debs
   moon:~/debs# cp /usr/src/*.deb .
   ```

3. A vytvoříme soubor `Packages`

   ```
   moon:~/debs# dpkg-scanpackages . /dev/null >Packages
   ```

4. Do souboru `/etc/apt/sources.list` pak přidáme nově vytvořený zdroj balíčků. Jedná se o řádek

   ```
   deb file:/root/debs ./
   ```

5. A nyní můžeme připravené balíčky instalovat

   ```
   moon:~# apt-get install kernel-image-2.4.18
   ```

Jiný postup se objevil jako příspěvek v konferenci Debian Backports

> In a directory with downloaded deb packages which you want to include in
> the local APT repo, make an empty file "Override", cd into the directory and run
>
> dpkf-scanpackages . Override >Packages
>
> and put this line in your /etc/apt/sources list
>
> deb file:/path/to/your/repo ./

6.9.16.5. Automatický noční upgrade

Automatický upgrade v stanovenou dobu zajistíme načasováním programem `cron`. Pomocí příkazu

```
# crontab -e
```

upravíme soubor s úlohama tak, že přidáme či modifikujeme řádky

```
PATH=/bin:/usr/bin:/sbin:/usr/sbin

0 1 * * *       apt-get update&& apt-get upgrade
```

6.9.16.5.1. Načasování a pravidelná upgrade s použitím cron-apt

Zdroje a odkazy:

- Balíčkovací systém distribuce Debian GNU/Linux, část šestá (http://root.cz/clanek.php4?id=1769)

ToDo list

1. FIXME:

Místo zcela vlastního ručního způsobu můžeme využít existující balíček, `cron-apt`. Ve Woody je ve verzi `0.0.6woody1`. Přináší nám více možností jak si automatický upgrade dokonfigurovat a přizpůsobit. Nainstalujeme jej:

```
# apt-get install cron-apt
```

Po nainstalování je potřeba si dokonfigurovat cron-apt k obrazu svému. Konfigurace je uložena v adresáři `/etc/cron-apt`. Zde se nachází konfigurační soubor `/etc/cron-apt/config`:

```
MAILTO=radek.hnilica@firma.cz
MAILON=upgrade
```

a adresář s akcemi `/etc/cron-apt/action.d`. Tento obsahuje dvě standardní akce `/etc/cron-apt/action.d/0-update`:

```
update
```

a `/etc/cron-apt/action.d/3-download`:

```
upgrade -d -y
autoclean -y
```

6.9.17. Aptitude

Poznámka: Aptitude má ve standardní konfiguraci jednu nešťastně nastavenou volbu. Jedná se o `Recommends-Important` jenž je nastavena na true. To způsobuje že aptitude instaluje v rámci závislostí i balíčky jenž jsou jen doporučené. Tím pádem sa časo nainstaluje software který vysloveně nechceme instalovat. Toto chování opravíme v konfiguraci nastavením:

```
// Konfigurace Aptitude
Aptitude {
        Recommends-Important false;     // Neinstaluj doporucene balicky
}
```

6.9.18. Neobvyklý hardware

* *section id="specialni-instalace" xreflabel="Neobvyklý hardware"*
* *rcsindo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Zdroje a odkazy:

- Software Raid (http://www.wlug.org.nz/SoftwareRaid)
- .

ToDo list

1. FIXME:

V této sekci popisuji instalace které neprobíhaly standardním způsobem.

6.9.18.1. Instalace na ataraid PDC20267 s mirroringem

Zdroje a odkazy:

- ataraidhowto (http://people.redhat.com/arjanv/pdcraid/ataraidhowto.html)
- Instalar Debian en un RAID Promise PDC202xx en 20 minuts (http://bulma.net/body.phtml?nIdNoticia=1067)
- Re: Promise Fasttrak 100 PDC20267 on Debian (http://linux.derkeiler.com/Mailing-Lists/Debian/2003-10/2765.html)
- Re: Promise Fasttrak 100 PDC20267 on Debian (http://linux.derkeiler.com/Mailing-Lists/Debian/2003-10/2798.html)
- .

V připreveném jádře musíme mít zapnuty tyto volby

```
CONFIG_BLK_DEV_HPT366         HPT366 chipset support.
CONFIG_BLK_DEV-PDC202X        PROMISE PDC202 {46|62|65|67|68} support
CONFIG_PDC202XX_FORCE         Special FastTrack Feature
CONFIG_BLK_DEV_ATARAID        Support for IDE Raid Controllers
CONFIG_BLK_DEV_ATARAID_PDC    Support Promise software RAID (Fasttrak(tm))
CONFIG_BLK_DEV_ATARAID_HPT    Highpoint 370 software RAID

/IDE, ATA and ATAPI Block devices
        <*> HPT36X/37X chipset support
        <*> PROMISE PDC202{46|62|65|67} support
        [*]  Special UDMA Feature
        <*> Support for IDE Raid controllers (EXPERIMENTAL)
        <*>   Support Promise software RAID (Fasttrak(tm)) (EXPERIMENTAL)
        <*>   Highpoint 370 software RAID (EXPERIMENTAL)
```

Při bootu by se mělo ukázet něco takového

```
Uniform Multi-Platform E-IDE driver Revision: 7.00beta4-2.4
ide: Assuming 33MHz system bus speed for PIO modes; override with idebus=xx
PIIX4: IDE controller at PCI slot 00:07.1
PIIX4: chipset revision 1
PIIX4: not 100% native mode: will probe irqs later
    ide0: BM-DMA at 0xf000-0xf007, BIOS settings: hda:pio, hdb:pio
    ide1: BM-DMA at 0xf008-0xf00f, BIOS settings: hdc:pio, hdd:pio
PDC20267: IDE controller at PCI slot 00:08.0
PCI: Found IRQ 10 for device 00:08.0
PDC20267: chipset revision 2
PDC20267: not 100% native mode: will probe irqs later
PDC20267: ROM enabled at 0xe4000000
PDC20267: (U)DMA Burst Bit ENABLED Primary PCI Mode Secondary PCI Mode.
    ide2: BM-DMA at 0xe400-0xe407, BIOS settings: hde:DMA, hdf:DMA
    ide3: BM-DMA at 0xe408-0xe40f, BIOS settings: hdg:DMA, hdh:DMA
hda: ST38410A, ATA DISK drive
blk: queue c02b9140, I/O limit 4095Mb (mask 0xffffffff)
```

```
hdc: IBM-DTLA-307045, ATA DISK drive
blk: queue c02b9594, I/O limit 4095Mb (mask 0xffffffff)
hde: ST380023A, ATA DISK drive
hdf: ST380021A, ATA DISK drive
blk: queue c02b99e8, I/O limit 4095Mb (mask 0xffffffff)
blk: queue c02b9b24, I/O limit 4095Mb (mask 0xffffffff)
hdg: IC35L040AVER07-0, ATA DISK drive
hdh: ST340810A, ATA DISK drive
blk: queue c02b9e3c, I/O limit 4095Mb (mask 0xffffffff)
blk: queue c02b9f78, I/O limit 4095Mb (mask 0xffffffff)
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
ide2 at 0xd400-0xd407,0xd802 on irq 10
ide3 at 0xdc00-0xdc07,0xe002 on irq 10
hda: attached ide-disk driver.
hda: host protected area => 1
hda: 16841664 sectors (8623 MB) w/512KiB Cache, CHS=8354/32/63, UDMA(33)
hdc: attached ide-disk driver.
hdc: host protected area => 1
hdc: 90069840 sectors (46116 MB) w/1916KiB Cache, CHS=89355/16/63, UDMA(33)
hde: attached ide-disk driver.
hde: host protected area => 1
hde: 156301488 sectors (80026 MB) w/2048KiB Cache, CHS=155061/16/63, UDMA(100)
hdf: attached ide-disk driver.
hdf: host protected area => 1
hdf: 156301488 sectors (80026 MB) w/2048KiB Cache, CHS=155061/16/63, UDMA(100)
hdg: attached ide-disk driver.
hdg: host protected area => 1
hdg: 80418240 sectors (41174 MB) w/1916KiB Cache, CHS=79780/16/63, UDMA(100)
hdh: attached ide-disk driver.
hdh: host protected area => 1
hdh: 78165360 sectors (40021 MB) w/2048KiB Cache, CHS=77545/16/63, UDMA(100)
Partition check:
 /dev/ide/host0/bus0/target0/lun0: p1 p2 p3 p4
 /dev/ide/host0/bus1/target0/lun0: [PTBL] [5606/255/63] p1
 /dev/ide/host2/bus0/target0/lun0: p1
 /dev/ide/host2/bus0/target1/lun0: p1
 /dev/ide/host2/bus1/target0/lun0: [PTBL] [5005/255/63] p1
 /dev/ide/host2/bus1/target1/lun0: [PTBL] [4865/255/63] p1

PDC20267: IDE controller at PCI slot 02:0e.0
PCI: Found IRQ 9 for device 02:0e.0
PDC20267: chipset revision 2
PDC20267: not 100% native mode: will probe irqs later
PDC20267: (U)DMA Burst Bit ENABLED Primary MASTER Mode Secondary MASTER Mode.
    ide1: BM-DMA at 0xdf00-0xdf07, BIOS settings: hdc:pio, hdd:pio
    ide2: BM-DMA at 0xdf08-0xdf0f, BIOS settings: hde:pio, hdf:pio
hda: TEAC CD-552E, ATAPI CD/DVD-ROM drive
hdc: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c031eb0c, I/O limit 4095Mb (mask 0xffffffff)
hde: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c031ef78, I/O limit 4095Mb (mask 0xffffffff)
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0xdff0-0xdff7,0xdfe6 on irq 9
ide2 at 0xdfa8-0xdfaf,0xdfe2 on irq 9
hdc: host protected area => 1
hdc: 234441648 sectors (120034 MB) w/8192KiB Cache, CHS=14593/255/63, UDMA(100)
```

```
hde: host protected area => 1
hde: 234441648 sectors (120034 MB) w/8192KiB Cache, CHS=14593/255/63, UDMA(100)
hda: ATAPI 52X CD-ROM drive, 128kB Cache, UDMA(33)
Uniform CD-ROM driver Revision: 3.12
Partition check:

# cat /proc/ide/pdc202xx

PROMISE Ultra series driver Ver 1.20.0.7 2002-05-23 Adapter: Ultra100
--------------- Primary Channel ---------------- Secondary Channel
-------------
                enabled                         enabled
66 Clocking     disabled                        disabled
Mode            MASTER                          MASTER
--------------- drive0 --------- drive1 -------- drive0 ----------
drive1 ------
DMA enabled:    yes             yes             yes               yes
UDMA Mode:      5               5               5                 5
PIO Mode:       4               4               4                 4
```

Un fitxer amb permís d'execució que anomenarem 'makedev' (preferiblement en minúscules per distingir-lo del que es pot trobar a http://people.redhat.com/arjanv/pdcraid/ataraidhowto.html amb el següent contingut: `/floppy/makedev`

```
#!/bin/sh
    mkdir -p /target/dev/ataraid



    cd /target/dev/ataraid
    mknod d0 block 114 0
    mknod d0p1 block 114 1
    mknod d0p2 block 114 2
    mknod d0p3 block 114 3
    mknod d0p4 block 114 4
    mknod d0p5 block 114 5
    mknod d0p6 block 114 6
    mknod d0p7 block 114 7
    mknod d0p8 block 114 8
    mknod d0p9 block 114 9
    mknod d0p10 block 114 10
    mknod d0p11 block 114 11
    mknod d0p12 block 114 12
    mknod d0p13 block 114 13
    mknod d0p14 block 114 14
    mknod d0p15 block 114 15
```

As far as I know most distributions today (and definitely Debian Woody) use a utility called mdadm to administer SoftwareRaid devices on a Linux system. mdadm can be obtained from

One major issue for me is that you can use, and mount both the dev/ataraid/d0 devices, and the /dev/hd devices. This makes for lots of fun in the Red Hat installer, and Cerberus.

Stupidity management for the superuser is a user space issue in Unix systems. If the RH installer does let you do stupid things, please bugzilla it.

```
ICH2: IDE controller at PCI slot 00:1f.1
ICH2: chipset revision 5
ICH2: not 100% native mode: will probe irqs later
    ide0: BM-DMA at 0xffa0-0xffa7, BIOS settings: hda:DMA, hdb:pio
PDC20267: IDE controller at PCI slot 02:0e.0
PCI: Found IRQ 9 for device 02:0e.0
PDC20267: chipset revision 2
PDC20267: not 100% native mode: will probe irqs later
PDC20267: (U)DMA Burst Bit ENABLED Primary MASTER Mode Secondary MASTER Mode.
    ide1: BM-DMA at 0xdf00-0xdf07, BIOS settings: hdc:pio, hdd:pio
    ide2: BM-DMA at 0xdf08-0xdf0f, BIOS settings: hde:pio, hdf:pio
hda: TEAC CD-552E, ATAPI CD/DVD-ROM drive
hdc: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c031eb0c, I/O limit 4095Mb (mask 0xffffffff)
hde: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c031ef78, I/O limit 4095Mb (mask 0xffffffff)
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0xdff0-0xdff7,0xdfe6 on irq 9
ide2 at 0xdfa8-0xdfaf,0xdfe2 on irq 9
hdc: host protected area => 1
hdc: 234441648 sectors (120034 MB) w/8192KiB Cache, CHS=14593/255/63, UDMA(100)
hde: host protected area => 1
hde: 234441648 sectors (120034 MB) w/8192KiB Cache, CHS=14593/255/63, UDMA(100)
hda: ATAPI 52X CD-ROM drive, 128kB Cache, UDMA(33)
Uniform CD-ROM driver Revision: 3.12
Partition check:
 hdc: hdc1 hdc2 hdc3 hdc4 < hdc5 >
 hde: hde1 hde2 hde3 hde4 < hde5 >
Promise Fasttrak(tm) Softwareraid driver 0.03beta: No raid array found
Highpoint HPT370 Softwareraid driver for linux version 0.01
No raid array found
SCSI subsystem driver Revision: 1.00
```

```
/dev/ataraid/dX
/dev/ataraid/dXpY
```

```
lilo.conf:
boot=/dev/ataraid/d0p3
root=/dev/ataraid/d0p3

default=Linux
disk=/dev/ataraid/d0
 bios=0x80
#
image=/boot/vmlinuz
 label=Linux
  read-only
```

Při bootu z bf2.4 diskety se objeví

```
PDC20267: IDE controller at PCI slot 02:0e.0
```

```
PCI: Found IRQ 9 for device 02:0e.0
PDC20267: chipset revision 2
PDC20267: not 100% native mode: will probe irqs later
PDC20267: (U)DMA Burst Bit ENABLED Primary MASTER Mode Secondary MASTER Mode.
    ide1: BM-DMA at 0xdf00-0xdf07, BIOS settings: hdc:pio, hdd:pio
    ide2: BM-DMA at 0xdf08-0xdf0f, BIOS settings: hde:pio, hdf:pio
hda: TEAC CD-552E, ATAPI CD/DVD-ROM drive
hdc: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c031eb0c, I/O limit 4095Mb (mask 0xffffffff)
hde: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c031ef78, I/O limit 4095Mb (mask 0xffffffff)
Uniform CD-ROM driver Revision: 3.12
Partition check:
 hdc: hdc1 hdc2 hdc3 hdc4 < hdc5 >
 hde: hde1 hde2 hde3 hde4 < hde5 >
...



PDC20627: IDE controller at PCI slot 02:0e.0
PDC20627: chipset revision 2
PDC20627: not 100% native mode: will probe irqs later
PDC20627: (U)DMA Burst Bit ENABLED Primary MASTER Mode Secondary MASTER Mode.
    ide2: BM-DMA at 0xdf00-0xdf07, BIOS settings: hdc:pio, hdd:pio
    ide2: BM-DMA at 0xdf08-0xdf0f, BIOS settings: hde:pio, hdf:pio
hda: TEAC CD-552E, ATAPI CD/DVD-ROM drive
hde: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c03426d8, I/O limit 4095Mb (mask 0xffffffff)
hdg: WDC WD1200JB-00EVA0, ATA DISK drive
blk: queue c0342b44, I/O limit 4095Mb (mask 0xffffffff)
ide0: at 0x1f0-0x1f7,0x3f6 on irq 14
ide2: at 0xdff0-0xdff7,0xdfe6 on irq22
ide3: at 0xdfa8-0xdfaf,0xdfe2 on irq22
hde: attached ide-disk driver.
hde: 234441648 sectors (120034MB) w/8192KiB Cache, CHS=14593/255/63, UDMA(100)
hdg: attached ide-disk driver.
hdg: 234441648 sectors (120034MB) w/8192KiB Cache, CHS=14593/255/63, UDMA(100)
hda: attached ide-cdrom driver.
hda: ATAPI 52X CD-ROM drive, 128kB Cache, UDMA(33)
Uniform CD-ROM driver Revision: 3.12
Partition check:
 hde: hde1 hde2 hde3 hde4 < hde5 >
 hdg: hdg1 hdg2 hdg3 hdg4 < hdg5 >
 ataraid/d0: ataraid/d0p1 ataraid/d0p2 ataraid/d0p3 ataraid/d0p4 < ataraid/d0p5 >
Drive 0 is 114473 Mb (33 / 0)
Drive 1 is 114473 Mb (34 / 0)
Raid1 array consists of 2 drives.
Promise Fasttrack(tm) Softwareraid driver for linux version 0.03beta
...
VFS: Cannot open root device "" or 03:02
Please append a correct "root=" boot option
```

V GRUBu zadám

```
grub> root (hd0,1)
grub> kernel (hd0,1)/boot/vmlinuz-2.4.25-ariane root=/dev/ataraid/d0p2
```

```
grub> boot
```

6.9.18.2. Instalace na ZIPku

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Instalace na ZIPku, tedy taková instalace, kdy je systém startován ze ZIPky a na této je také kořenový souborový systém (root).

> **Varování**
>
> ZIP mechaniku je třeba „naswitchovat“ jako disketovou mechaniku, nikoli pevný disk! S nastavením ZIPky jako pevného disku jsem su užil trápení. Nakonec se mi ji podařilo v jednom počítači (DELL PowerEdge 350 což je INTEL ISP 1100) sprovoznit s pomocí FreeDosu.

Nejdříve si připravíme ZIPku. Já ji mám jako zařízení `/dev/hdc`. Pomocí programu fdisk vytvořím oddíl `/dev/hdc1`. Poté na tomto oddílu vytvořím souborový systém:

Příklad 6-6.

```
# mkfs -c /dev/hdc1
```

 # mkfs -c /dev/hdc1 Tím mám ZIPku připravenu. Vsunu ji do počítače ve kterém ji budu instalovat a zasunu rovněž upravenou [rescue] disketu. Počitač restartujeme.

Později jsem měl na uvedeném routeru problém se síťovou kartou. Byl to problém technický, špatně zasunutá karta, ale než jsem na to došel, tak jsem předělal schéma disku. Nejdříve jsem zkoušel jádru předávat parametry. Protože jsem samozřejmě neuspěl, přepracoval jsem bootování. Zipka je jeden veliký disk vytvořený takto

```
# mkfs /dev/hdc
# mount /dev/hdc /mnt/zip
# cp -a root/* /mnt/zip
# cp -a host/hostname/* /mnt/zip
# chroot /mnt/zip lilo -v
# umount /mnt/zip
```

Ted ze ZIPky bootuji přímo jádro bez nutnosti použít FreeDos.

6.9.18.3. Servery DELL

Vzhledem k hardware serverů DELL je nejlepší volbou stáhnou obraz instalčního CD pro tyto servery ze stránky Debian on Dell Servers (http://wiki.osuosl.org/display/LNX/Debian+on+Dell+Servers). Obraz je malý a slouží jen k nabootování stroje s ovladači pro hardware serverů DELL a spuštění instalace. Tu dále provádím ze síťe.

Při startu s tohoto CD máme možnost jak spustit instalaci, volba `linux`, tak použít CD jako záchranné, volba `rescue root=/dev/hd?`. Dále můžeme použít tyto argumenty:

- `quiet` —
- `verbose` —
- `debug` —
- `nolangchooser` —

- `bootkbd` —

Ukázka spuštění:

```
: linux verbose
```

6.9.19. Nezpracováno / nezařazeno

* *section condition="author"*
* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

6.9.19.1. Vytvoření startovací diskety

```
# mkboot path_to_kernel

# mkboot boot/vmlinuz-2.4.18-bf2.4
```

6.10. Zavaděče jádra

Zavaděče jádra jsou program jejichž hlavním cílem je, jak již z názvu vyplývá, zavést jádro linuxu do paměti a předat mu řízení. Součástí této operace bývá i zavádění ramdisku (*initrd*) s moduly. Zejména v novějších distribucích se postupně přechází právě od jádra monolitickému k „jádru modulárnímu“.

Podívejme se tedy na jednotlivé zavaděče blíže.

6.10.1. lilo

LILO je jedním ze starších zavaděčů.

```
/sbin/lilo -q
/sbin/lilo -R Linux
/sbin/lilo -I Linux
/sbin/lilo {-u|-U}
```

6.10.2. lodlin

Tento zavaděč běží v prostředí DOSu.

6.10.3. grub

Odkazy:

- Reinstalling GRUB (http://www.youtube.com/watch?v=WtBBl6HvdpM)

Grub je s nadsázkou rečeno vlastní operační systém, jenž slouží pro zavádění (jader) operačních systémů. Umí zavádět všechny běžné OS jako jsou Linux, (BSD), MS Windows.

Všechny soubory grubu se nachází v adresáři `/boot/grub`, a to i konfigurační soubor. Tedy nikoliv v `/etc/grub/...` jak by člověk očekával.

Význam některých souborů:

`device.map`

Mapa zařízení zjišťovaná autodetekcí. Vytváří grub při instalaci.

(hd0) /dev/hda
(fd0) /dev/fd0

`menu.lst`

FIXME: Soubor popisuje konfiguraci grubu.

```
color white/blue black/light-gray
default 0
timeout 8

title Linux
    root (hd0,2)
    kernel /boot/vmlinuz root=/dev/hda3
    initrd /boot/initrd
```

GRUB `/boot/grub` `menu.lst` (u RH `grub.conf`) `(hd0,1)` disk první partition 2 většinou hda2

```
title Linux
 kernel (hd0,1)/boot/vmlinuz root=/dev/hda2
 initrd (hd0,1)/boot/initrd
```

`linux.rc` - startovací skript

```
grub> root (hd0,TAB
  Possible partitions are:
  ...

grub> cat /etc/fstab
grub> kernel /boot/vmlinuz root=/dev/hda2
grub> initrd /boot/initrd.img

grub> setup (hd0)
grub> boot

grub> pager
```

Některé příkazy Grubu:

kernel

Jádro a parametry jádra.

```
kernel /boot/vmlinuz root=/dev/hda3
```

initrd

Init RAM disk. Tímto parametrem oznámíme grubu kde se nachází případný initramdisk daného jádra jenž jsem vybrali příkazem **kernel**.

root

Vybrání disku a oddílu který se bude používat při všech operacích jako kořenový.

boot

Zavedení aktuálně vybraného jádra (kernel) se všemi dalšími volbami.

FIXME:

FIXME:

* ***FIXME:****Popsat příkazy:* ***root****,*

Poznámka: Grub umí na stisk klávesy TAB doplňování dle kontextu. V případě že není možno doplnit, grub vypíše možné volby/texty.

Použití grubu při bootování do single user režimu.

```
kernel /boot/vmlinuz 1 root=/dev/hda3 ...
```

Po zavedení jádra přejde systém do init režimu 1 (single user). A vyzve nás k zadání hesla uživatele root. Pokud chceme obejít i toto, použijeme parametr *init* jímž místo programu **init** jenž se spouští defaultně, spustíme například shell **/bin/sh**.

```
kernel /boot/vmlinuz 1 init=/bin/sh root=/dev/hda3 ...
```

Nastavení sériové linky

```
# Nastaveni seriove konzole
serial --unit=0 --speed=38400 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console
kernel console=ttyS0,38400 serial=0,38400n8 console=tty1,38400
```

Zavádění alternativních systémů.

```
title Windows
    root (hd0,0)
    chainloader +1
```

Spuštění testu paměti.

```
title MemTest86
root (fd0)
kernel /boot/memtest86.bin
```

Instalace grubu z grubu

```
title Obnova zavadece SLES9 + Boot
    root (hd0,2)
    setup (hd0)         # ...
    kernel /boot/vmlinuz root=/dev/hda3
    initrd /boot/initrd
```

6.10.3.1. Instalace

```
# apt-get install grub

# grub-install --root-directory=/boot /dev/hda
```

```
silo:~# grub-install --root-directory=/ /dev/hda
Probing devices to guess BIOS drives. This may take a long time.
end_request: I/O error, dev 02:00 (floppy), sector 0
Installation finished. No error reported.
This is the contents of the device map //boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script ‘grub-install’.

(fd0)   /dev/fd0
(hd0)   /dev/hda
silo:~# _
```

```
(fd0) /dev/fd0
(hd0) /dev/hda
```

Pokus o instalaci grubu na disketu

```
# grub-install /dev/fd0
```

Nevyšlo to. Správný postup podle dokumentace má být

```
# cd /usr/lib/grub/i386-pc
# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
# dd if=stage2 of=/dev/fd0 bs=512 seek=1
186+1 records in
186+1 records out
```

6.10.3.1.1. Instalace na disketu

Přípravíme si disketu. Nejdříve ji naformátujeme programem **format** (**superformat**)

```
# fdformat /dev/fd0
??? # mkfs.msdos /dev/fd0
```

Nebo **superformat**

```
# superformat FIXME:
```

FIXME:

```
# mount /media/floppy
# mkdir -p /media/floppy/boot/grub
```

```
# cp /usr/lib/grub/* /media/floppy/boot/grub/
# grub-install --root-directory=/media/floppy /dev/fd0
Probing devices to guess BIOS drives. This may take a long time.
Installation finished. No error reported.
This is the contents of the device map /media/floppy/boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script ‘grub-install’.

(fd0)   /dev/fd0
(hd0)   /dev/hda
# cp /boot/grub/menu.lst /media/floppy/boot/grub/
```

Varování

Pozor na jádro řady 2.6.x! Zde je automounter jenž koliduje s **grub-install**. V tom případě je třeba automounter vypnout `umount /media/floppy`, připojit disketu do jiného adresáře `mount /dev/fd0 /mnt` a spustit **grub-install** takto:

```
# grub-install --root-directory=/mnt /dev/fd0
```

6.10.4. grub2

Fakta:

- Je použit jako defaultní zavaděč v Ubuntu od verze 9.10 (Karmic)

Grub 2 se liší od grubu 1.x. Hlavní konfigurační soubor který grub2 používá pří startování počítače se jmenuje `/boot/grub/grub.cfg`. Tento soubor se neupravuje ručně, ale je generován automaticky, programem **/usr/sbin/grub-mkconfig** podle informací v souboru `/etc/default/grub` a informací v adresáři `/etc/grub.d`.

Kapitola 7. Disky

Disky a další paměťová média

* *chapter id="disky"*

7.1. Diskety

Formátování diskety:

```
# fdformat /dev/fd0H1440
```

nebo

```
# setfdprm /dev/fd0 1440/1440
# fdformat /dev/fd0
```

Pro formátování disket je nově doporučeno používat program superformat, který dokáže naformátovat disketu na „vyšší“ kapacitu.

Obyčejnou disketu (3.5", 1.44MB) sformátujeme například takto

```
# superformat -V --superverify /dev/fd0 hd
```

7.2. Disky

Vytvoření souborového systému

```
# mkfs -c -v /dev/disk
```

Takto vytvoříme obecně souborový systém. Standardně se vytvoří `ext2`. Pokud chceme podrobnější nastavení, voláme přímo program mke2fs. Například u velikých disků je 5% kapacity disku vyhrazených správci často zbytečně mnoho, a tak si pomocí přepínače `-m` řekneme o míň, například jen 1%

```
# mke2fs -c -m 1 /dev/hda3
```

7.2.1. Diskové oddíly

Disk rozdlíme na oddíly příkazem **fdisk** nebo **cfdisk**.

7.2.2. Vytvoření a inicializace oddílu pro swapování

Swapovací oddíl se vytváří jako kterýkoliv jiný. Nevytváříme však na něm souborový systém, ale provádíme jeho inicializaci. Například inicializace swapovacího oddílu na `/dev/hda3` se provede jednoduše takto

```
# mkswap -c /dev/hda3
```

7.2.3. Vytvoření souborového systému na disku s vadnými bloky

Nejdříve pořídíme seznam vadných bloků

```
# badblocks /dev/hdxx velikost_disku -o badblock.txt
```

Pokud je disk již zformátovaný, pouze oznámíme filesystému které bloky nemá používat

```
# fsck -t ext2 -l badblock.txt /dev/hdxx
```

Při formátování to provedeme takto

```
# mkfs -t ext2 -l badblock.txt /dev/hdxx
```

7.2.4. Vytvoření souborového systému na velikém disku

Při vytváření souborového systému na velikém disku je třeba si uvědomit, že některé typy souborových systémů, například `ext2` standardně vyhrazují 5% prostoru pro administrátora. Na velikém disku tvoří pak takto vahrazený prostor značnou část kapacity.

```
# mke2fs -m 0.1 /dev/hda1
```

7.2.5. Nastavování a optimalizace parametrů IDE disku

Po nainstalování balíčku hwtools je vhodné si přečíst soubor `/usr/share/doc/hwtools/README.Debian`

* *Vypsáno z článku IDE Disky na pokraji šialenstva (http://www.linuxzone.cz/index.phtml?ids=29&idc=419)*

Příklad 7-1. Skript `hdtest`

```
#!/bin/sh
# $Id: hdtest,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $
# $Source: /home/radek/cvs/unix-book/example/script/hdtest,v $
# Optimalizace nastaven<65533> IDE disku

LOG=hdtest.log
cmd="hdparm -q -d $d -X $X $1"

rm -f $LOG

for d in 0 1; do
 for X in 33 66 100 133; do
  for p in 1 2 3 4 5; do
```

```
      for a in 4 8 16; do
       for c in 0 1 3; do
        for m in 0 2 4 8 16; do
         for u in 0 1; do
          for W in 0 1; do
           cmd="hdparm -q -d $d -X $X -p $p -A1 -a$a -c$c -m$m -u$u -W$W $1"
    echo $cmd
    #$cmd >/dev/null
    #hdparm -q -d $d -X $X >/dev/null
    echo "Nastavenie: $cmd" | tee -a $LOG
    #hdparm -tT $1 | tee -a $LOG
    echo "----------------------" | tee -a $LOG
          done
         done
        done
       done
      done
     done
    done
   done
```

Nainstalujeme balíčky

```
# apt-get install hwtools hdparm
```

a provedema jednoduchou optimalizaci. Nejdříve se podíváme jak je disk nastaven

```
# hdparm /dev/hda
```

poté provedem jednoduché měření

```
# hdparm -tT /dev/hda
```

a zkusíme nastavit některé parametry, například

```
# hdparm -c1 -W1 -X66 -m16 /dev/hda
```

vazkoušíme různé hodnoty těchto i jiných parametrů a jejich kombinace. Výsledné nastavení se kterým jsme spokojeni zapíšeme do souboru `/etc/init.d/hwtools` aby se uplatnilo po každém spuštění počítače.

7.3. Archivování disket (obrazů disket)

Sepsáno podle Archivink Floppy Disks Using Images (http://sharkysoft.com/tutorials/notes/linux/floppyimages/)

Někdy je potřeba archivovat po delší dobu diskety. Vzhledem k problémům které jsou s tímto médiem js mnohem bezpečnější uložit si obsahy disket na jiné médium, například CD-R. Nejednodušší je uložit obsah celé diskety sektor po sektoru. Pak nás nemusí zajímat souborový systém na disketě, a můžeme tak archivovat diskety jejichž souborový systém není znám. Obsah 3½" diskety uložíme do souboru `floppy_image` takto

```
# dd if=/dev/fd0 of=floppy_image count=1 bs=1440k
```

Obnovení diskety s příslušného obrazu diskety pak uděláme v UNIXu jednoduše obdobným způsobem. Například výše míněný obraz obnovíme na disketu následovně

```
# dd of=floppy_image if=/dev/fd0 count=1 bs=1440k
```

Takto vytvořený obraz diskety má výhodu v tom, že k obnovení diskety můžeme použít i jiný nástroj. Například pod DOSem je to program rawrite.exe jenž bývá součástí instalace distribucí Linuxu.

7.4. Správa souborového systému

7.4.1. Nouzový stav

Může se nám stát, že při nekorektním ukončením práce, například výpadkem proudu dojde k „poškození“ záznamů na disku. Systém pak po nabootování „naběhne“ do nouzového stavu/režimu (*emergency mode*). Pomůže nám pak podobný postup jako zde uvedený.

```
# fsck /
# mount -n -o rw,remount /      # přemountování / s právem pro zápis
# fsck other_partitions
# mount -a
```

7.5. Souborové systémy

Seznam souborových systémů

- minix
- xiafs
- ext2
- ext3
- ReiserFS

7.5.1. ext2

FIXME:

```
# mkfs.ext2 /dev/hda5
```

7.5.2. ReiserFS

* ***FIXME:***

  Pomocné programy jsou v balíčku `reiserfsprogs`.

```
# mkreiserfs /dev/hda9
```

7.5.2.1. Převedení adresářů na ReiserFS

Někdy se vyskytne potřeba převést data ze stávajícího filesystému na žurnálovací, například ReiserFS. Buď přidáváme do stávajícího stroje nové disky, nebo při instalaci nemá instalační jádro podporu ReiserFS. Zejména druhý případ je v této době poměrně častý. Standardní instalační diskety jsou postaveny na jádru řady 2.2.x bez podpory žurnálovacích systémů. Pokud z nějakého důvodu nemáme možnost zavést systém z BF disket, můžeme použít následující postup. Při instalaci si naplánujeme rozdělení diskového prostoru na jednotlivé oddíly například takto:

```
hda1    /boot
hda2    / + /usr
hda3    swap
hda5    /var
hda6    /home
```

části `hda5` a `hda6` neininicializujeme a necháme v průběhu instalace adresáře `/var` a `/home` na části `hda2`. Po úspěšném ukončení základní instalace vyměníme jádro za jádro s podporou ReiserFS. Buď nainstalujeme standardní jádro 2.4.18, nebo si přeložíme vlastní jádro v řadě 2.4.x. Nezapomeneme přeložit podporu ReiserFS přímo do jádra. Jakmile máme systém běžící na jádře 2.4.x můžeme přistoupit k příprave neinicializovaných svazků. Nainstalujeme si nástroje

```
# apt-get install reiserfsprogs
```

a zinicializujeme části disku

```
# mkreiserfs /dev/hda5
# mkreiserfs /dev/hda6
```

Do `/etc/fstab` si připravíme záznamy popisující nové svazky `/home` a `/var`

```
/dev/hda5       /var            reiserfs defaults               0       3
/dev/hda6       /home           reiserfs defaults               0       4
```

Až do tohot okamžiku jsme svou činností žádným způsobem neomezovali případné pracující uživatel. Zbytek práce ovšem musíme provést v jednouživatelském režimu. Důvodem je potřeba zajistit si, že nikdo nebude pracovat se soubory v době kdy je budeme převádět na nové svazky.

```
# init s
```

Tak, systém je vyhrazen jen nám a my provedeme bez zbytečného otálení zbytek práce. Přejmenujeme adresář `/var`

```
# mv var var.old
```

Vytvoříme mount point pro nový `/var`

```
# mkdir var
```

připojíme k němu inicializovaný ReiserFS svazek

```
# mount /var
```

a překopírujeme do něj data z původního adresáře

```
# cp -a /var.old/* /var/
```

Stejným způsobem naložíme s adresářem `/home`. Když máme vše hotovo můžeme systém opět vrátit uživatelům přepnutím systému do víceuživatelského režimu

```
# init 2
```

Uschované data v adresářích `/var.sav` a `/home.sav` si můžeme ještě chvíli ponechat „pro stýčka příhodu“ a smazat je třeba až za několik dnů.

7.5.2.2. XFS

FIXME:

```
# mkfs.xfs -f /dev/hda5
```

7.5.3. Připojování svazků

FIXME:mount, **umount**, `/etc/fstab`, `/etc/mtab`

7.6. CryptoFS

* *section id="fragment.section_template" condition="author"*

Zdroje a odkazy:

- http://mail.nl.linux.org/linux-crypto/2002-03/msg00004.html

  Cryptoloop HOWTO (http://www.tldp.org/HOWTO/Cryptoloop-HOWTO/index.html)

  Disk Encryption HOWTO (http://www.tldp.org/HOWTO/Disk-Encryption-HOWTO/index.html)

  http://www.spinics.net/lists/crypto/msg00680.html

  The Linux CryptoAPI A User’s Perspective (http://www.kerneli.org/howto/) by David Bryson

  File Systems (http://www.linux-sec.net/FileSystem/)

  CD-Rom cryptes (http://linuxfr.org/~guyou/5925.html) — francouzsky

  Loop-Aes (http://plus-linux.de/wiki.cgi?Loop-Aes) — německy

  http://mail.nl.linux.org/linux-crypto/2002-07/msg00069.html

  Linux and cryptography (http://www.antipope.org/charlie/linux/shopper/167.crypto.html)

  USB-Memory-Sticks verschlüsseln mit GNU/Linux (http://www.uni-koblenz.de/~phil/linux/usbcrypt.html)

  Disk Encryption (http://encryptionhowto.sourceforge.net/Encryption-HOWTO-6.html)

  Using the Crypto File System (http://portal.suse.com/sdb/en/2001/06/jsj_crypto_filesystem_mini_howto.html)

  Loop-AES (http://sourceforge.net/projects/loop-aes/) on SourceForge

  . ()

  . ()

  . ()

ToDo list

1. FIXME:

FIXME:

In `/etc/conf.modules`, add:

```
alias loop-xfer-gen-0 loop_gen
alias loop-xfer-gen-10 loop_gen
alias cipher-4 blowfish            # Blowfish
alias cipher-6 idea                # IDEA
alias cipher-7 serp6f              # Serpent
alias cipher-8 mars6               # MARS
alias cipher-9 twofish             # Twofish
alias cipher-11 rc62               # RC6
alias cipher-15 dfc2               # DFC
```

Balíčky v Dabian/GNU Linux

- loop-aes-ciphers-source
- loop-aes-source
- loop-aes-utils

Example #4 from aespipe README: Create encrypted ISO9660 CD-ROM image that can be mounted using Linux loop-AES crypto package:

```
# mkisofs -r directory-tree | aespipe -e AES128 -T >image.iso
```

This image file can then be mounted under Linux like this:

```
# mount -t iso9660 image.iso /cdrom -o loop=/dev/loop0,encryption=AES128
```

Or, after writing image.iso to CD-ROM, like this:

```
# mount -t iso9660 /dev/cdrom /cdrom -o loop=/dev/loop0,encryption=AES128
```

aespipe is available here: http://loop-aes.sourceforge.net/aespipe/aespipe-v2.1c.tar.bz2

encrypted partition: (http://www.netsys.com/suse-linux-security/2003/06/msg00005.html)

```
# losetup /dev/loop0 /dev/hda4 --encryption aes -k 256 --phash sha512
enter your password
# mke2fs /dev/loop0
# mount /dev/loop0 /mnt
```

7.6.1. Kryptovaná CD

```
mkisofs -J -R -o cd3.img photo-album/
aespipe -e AES128 -T < cd3.img > cd3enc.img
cdrecord -v dev=0,0,0 -eject cd3enc.img
```

I could not mount it with ’mount’, but I’ve been able to do this:

```
 losetup -e aes -k 128 -P sha256 /dev/loop0 /dev/cdrom
mount -o ro -t iso9660 /dev/loop0 /cryptcd
```

Do `/etc/fstab` přidáme

```
/dev/cdrom /cryptcd iso9660 noauto,owner,kudzu,ro,loop=/dev/loop0,encryption=aes,keybits=128,p
```

7.7. Zálohování

FIXME:

7.7.1. Zálohování na pásky

FIXME:

Pro práci s páskou máme program **mt**.

```
sunlight:~# mt -f /dev/nst0 status
drive type = Generic SCSI-2 tape
drive status = 318767616
sense key error = 0
residue count = 0
file number = 0
block number = 0
Tape block size 512 bytes. Density code 0x13 (DDS (61000 bpi)).
Soft error count since last status=0
General status bits on (41010000):
 BOT ONLINE IM_REP_EN
```

Operace s páskou

eof
weof

Na aktuální pozici zapíše značku EOF.

rewind

Přetočí pásku na začátek média.

offline
rewoffl

Přetočení pásky na začátek a její vyhození z mechaniky, podporuje li mechanika tuto možnost.

status

Informace o stavu páskové mechaniky

retension

.

erase

.

datcompression

Ovládání hardwarové komprese na některých mechanikách SCSI-2 DAT. Zadáme-li jako parametr 1, vypíše se aktuální nastavení. Hodnota 0 vypne a jiná hodnota zapne kompresi.

```
sunlight:~# mt -f /dev/nst0 datcompression 0
Compression off.
Compression capable.
Decompression capable.
```

Jak je vidět komprese je vypnuta. Zapneme ji tedy.

```
sunlight:~# mt -f /dev/nst0 datcompression 2
Compression on.
Compression capable.
Decompression capable.
```

.

.

Ostatní operace: fsf, bsf, fsr, bsr, bsfm, fsfm, asf, eom, fss, bss, wset, eod, seod, setblk, setdensity, drvbuffer, stopoptions, stwrthreshold, seek, tell, densities, datacompression,

7.8. SW RAID

- Debian, (K)Ubuntu, Knoppix Notes (http://www.novaglobal.com.sg/?q=node/71)

```
# aptitude install mdadm
```

* ***FIXME:***

7.9. LVM

Odkazy:

- A simple introduction to working with LVM (http://www.debian-administration.org/articles/410)
- LVM HOWTO (http://www.tldp.org/HOWTO/LVM-HOWTO/)
- A Beginner's Guide To LVM (http://www.howtoforge.com/linux_lvm)
- Linux LVM Guide (http://www.linuxtopia.org/online_books/linux_lvm_guide/)
-

LVM — *Logical Volume Manager*, je vrstva mezi operačním systémem a fyzickými disky která vitváří virtuální svazky. Jejím základním úkolem je z řady fyzických disků vytvořit virtuální prostor z nehož je pak přidělováno místo logickým svazkům.

Vertikálně se tedy celý LVM dělí na tři oblasti.

lv	logical volumes	nejvyšší vrstva
vg	volume groups	střední vrstva
pv	physical volumes	nejnižší

Každá vrstva provádí jinou činnost a k jejímu ovládání/konfigurování slouží jiná skupina programů. Jména těchto programů začínají písmeny pv/vg/lv.

7.9.1. LVM a Debian Lenny

Odkazy:

- Debian Lenny and grub with /boot in LVM (http://jim.studt.net/depository/index.php/debian-lenny-and-grub-with-boot-in-lvm)

Poznámka: Při instalaci Debian Lenny z cd multiarch jsem si všiml jedné věci. Pokud už nějaké svazky mám, nainstaloval jsem si nejdříve do jedné primární partition rescue instalaci, a ve zbytku se pokusím vytvořit LVM pro produkční instalaci, nenabídne mi instalátor možnost použití grubu. Instalaci jen LVM na celý disk jsem nezkoušel.

Situaci jsem vyřešil tak, že jsem vytvořil malou partition `/boot` a vše ostatní dal do LVM. V takové konfiguraci se mi použití grubu již nabízelo.

7.9.2. Příklady a ukázky

7.9.2.1. Vytvoření physical volumes

```
# pvcreate /dev/hda5
# pvcreate /dev/hda7
```

7.9.2.2. Vytvoření skupiny `volume group`

```
# vgcreate main /dev/hda5
```

7.9.2.3. Přidání dalšího physical volume do volume groups

```
# vgextend main /dev/hda7
```

7.9.2.4. Vytvoření `logical volume`

```
# lvcreate -n src --size 6g main
  Logical volume "src" created
```

Vytvořený logický svazek můžeme rovnou inicializovat a připojit.

```
# mkfs.ext3 /dev/main/src
```

Po nějaké době se mi zaplnil disk `/`. Rozhodl jsem se z něj vyčlenit adresář `/usr`. Nejdříve si prohlédneme stávající skupiny, jestli na nich bude dost místa.

```
# vgdisplay
```

Nový svazek určený pro adresář `/usr` vytvořím ve skupině `main`.

```
# lvcreate -n usr --size 2g main
  Logical volume "usr" created
# mkfs.ext3 /dev/main/usr
```

7.9.2.5. Vyjmutí physical volume

```
# pvmove /dev/hda5 /dev/hda7
```

V případě přeřušení operace například restartem systému či jiným výpadkem musíme operaci dokončit. Zadáme příkaz pvmove, a ten rozpozná nedokončené operace a pokračuje v nich od posledního kontrolního bodu.

```
# pvmove -v
```

Po dokončení operace a uvlonění physical volume `/dev/hda5` jej vyjmeme ze skupiny.

```
# vgreduce main /dev/hda5
  Removed "/dev/hda5" from volume group "main"
```

7.10. Automatické připojování svazků

Odkazy:

- Quick autofs Tutorial (http://www.linuxhq.com/lg/issue24/nielsen.html) By Mark Nielsen
- Autofs Automounter HOWTO (http://www.linux-consulting.com/Amd_AutoFS/autofs.html)
- Automount mini-Howto (http://tldp.org/HOWTO/Automount.html)
- How to set up autofs (http://www.greenfly.org/tips/autofs.html)
-
-
-

Pokud nechceme pokaždé ručně připojovat svazky do systému musíme si tuto činnost nějak zautomatizovat. Tou nejjednodušší metodou, která funguje jen na svazky které jsou k dispozici při startu operačního systému a po celou dobu jeho činnosti je použití volby `auto` u příslušného svazku v souboru `/etc/fstab`. Tuto volbu použijeme při připojování svazků z lokálních disků. Její použití pro ostatní (výměnná) média a pro síťové svazky je velmi omezené a v řadě případů nevyhovující. Přesně pro tyto případy máme k dispozici jiné nástroje:

- amd — *automounter daemon*, starší řešení jenž beží v uživatelském prostoru a používá nfs
- autofs — nové řešní začleněné do jádra Linuxu

7.10.1. autofs

Autofs je program a balíček pro automatické připojování (mount) a odpojování (umount) výměnných médií a síťových svazků. Program je vylmi chytrý a předkonfigurovyný. Po odkomentování několika řádků v `/etc/auto.master` je plně funkční a můžeme jej používat. Nicméně „chytrost“ programu a jeho standardní konfigurace pro nás může být problémem. Před vlastní instalací tedy popíši jak všechno funguje a připravím si plán konfigurace. Teprve poté přistoupím k vlastní instalaci.

Poznámka: Autofs je implementován jako jaderný modul v Linuxu od verze 2.2.x.

Z vnejšího pohledu funguje autofs tak, že si vytvoří několik virtuálních adresářů do kterých podle potřeby připojuje a od kterých odpojuje svazky výměnných médií či síťové svazky. Uživatel například zadá příkaz

```
$ ls /smb/mirek/Data
```

Autofs podle konfigurace zjistí na kterém serveru je příslušný svazek a transparentně ho připojí v okamžiku potřeby, tedy v okamžiku běhu příkazu ls. Po delší době nepoužívání naopak síťový svazek odpojí. Uživateli tedy odpadne potřeba připojovat a odpojovat svazky, jakožto i nutnos znát podrobnosti jejich umístnění. Správce rovněž nemusí pro každý svazek zajišťovat konfiguraci v `/etc/fstab`.

Konfigurace autofs je dána hlavním konfiguračním souborem `/etc/auto.master` a mapovacími soubory či programy. Hlavní konfigurační soubor obsahuje jeden řádek pro každý „virtuální“ adresář, jako jsou například `/smb` a `/misc` v standardní konfiguraci. Na tomto řádku jseo specifikovány případné další volby a cesta k mapovacímu souboru či programu. Řádek konfiguračního souboru vypadá takto:

`/adresář` [volby] *`mapovací soubor`*

Pro doplnění, standardní konfigurace obsahuje řádky

```
/misc   /etc/auto.misc --timeout=60
#/smb   /etc/auto.smb
#/misc  /etc/auto.misc
#/net   /etc/auto.net
```

Autofs funguje tak, že převezme vládu nad několika adresáři podle konfigurace a tyto adresáře se pak chovají jako „okno“ do vyměnitelných médií či síťových svazků. Konfigurace konfigurace autofs a specifikování těchto adresářů je popsáno v hlavním konfiguračním souboru `/etc/auto.master`.

Struktura souboru je jednoduchá, soubor je řádkově orientovaný a na každém řádku je popis jednoho „automount adresáře“. Řádek má strukturu:

Bližší informace o struktuře souboru získáte příkazem `man 5 autofs`.

Podíváme se tedy na standardně vytvořený soubor.

Po odkomentování řádků s `/smb` a `/net` a reloadu konfigurace autofs můžeme začít vše používat.

```
# /etc/init.d/autofs reload
```

Jak to funguje? Autofs spustí pro každá konfigurační řádek v `/etc/auto.misc` proces, který vytvoří „virtuální“ adresář a načte si mapový soubor odpovídající tomuto adresáři. Při pokusu o přistoupení k podadresářům proces konzultuje mapový soubor a podle potřeby připojí příslušný zdroj.

A právě mapovací soubory jsou součástí chytrosti autofs. Můžeme totiž místo statických map popisujících mapování síťových adresářů použít skritpy které toto mapování sami vytváří. Soubory `/etc/auto.net` a `/etc/auto.smb` jsou toho ukázkou. Rozlišení mezi statickým mapovým souborem a skriptem dynamicky vytvářejícím mapu je v nastavení příznaku x (executable) těmto konfiguračním souborů.

V praxi to funguje tak, že pokud přistoupíme k souboru v adresářové struktuře `/smb`, autofs se podívá zdali má soubor /etc/auto.smb nastaven příznak x. Jestli ne, načte tento soubor jako mapu. Jestli ano, spustí jej a jako první parametr mu předá jméno podadresáře v /smb ke kterému přistupujeme. Skript /etc/auto.smb chápe toto jméno jako jméno serveru. Výstup skriptu /etc/aut.smb je pak chápán jako mapa. Toto "chytré" chování nám dovoluje automaticky připojovat svazky ze serverů o kterých jsme v době instalace autofs vůbec nevěděli.

7.10.1.1. Plánování konfigurace

7.10.1.2. Instalace a konfigurace

Nejdříve instalace. V Debian GNU/Linux Etch je balíček `autofs` který si nainstalujeme. Pokud nemáme nainstalováno, budeme možná potřebovat i následující dva balíčky. V těchto jsou programy které autofs potřebuje ke zjištění jaké síťové svazky dotazovaný síťový server exportuje.

```
# aptitude install autofs
# aptitude install nfs-common smbclient
```

Po instalaci je vytvořeno několik konfiguračních souborů, které si upravíme podle potřeby::

- `/etc/auto.master` — hlavní mapovací soubor
- `/etc/auto.misc`
- `/etc/auto.net`
- `/etc/auto.smb`

7.10.1.3. Moje konfigurace

Standardní konfigurace mi nevyhovovala, už proto že mám napsanou řadu skriptů které pracují se síťovými svazky a používají například adresář `/net`. Taky se mi nelíbilo rozdělení svazků samby a nfs do rozdílných adresářů. Přišle jsem tedy s těmito úpravami.

Konfiguraci pro nfs jenž je v souboru `/etc/auto.net` jsem přejmenoval na `/etc/auto.nfs`. Mohu tedy v hlavním konfiguračním souboru `/etc/auto.master` přejmenovat `net` na `nfs` a tím si uvolnit adresář net pro jiné použití.

```
# mv /etc/auto.net /etc/auto.nfs
# sed -e 's/net/nfs/g' /etc/auto.master >/etc/auto.master.new
# mv -f /etc/auto.master /etc/auto.master
```

Protože jsem chtěl použí adresář `/net` pro automatické připojení jak nfs tak samba svazků, vytvořil jsem mu konfigurační skript `/etc/auto.net` s následujícím obsahem.

```
#!/bin/sh
key=$1

# Ruční definice sít'ových svazků
cat <<EOF
-fstype=nfs,hard,intr,nodev,nosuid,nonstrict,async /root nas1:/mnt/HD_a2
EOF

# Mapa nfs svazků
/etc/auto.nfs $key

# Mapa samba svazků
/etc/auto.smb $key

# echo "/net /etc/auto.net" >>/etc/auto.master
```

7.10.2. Použití programu Amd

* *section id="fragment.amd" condition="author"*

  FIXME:TBD

Kapitola 8. Virtuální servery a emulátory

* *chapter id="virtual-server" xreflabel="Virtuální servery a emulátory"*

Odkazy a zdroje:

- Virtual private servers and security contexts (http://www.solucorp.qc.ca/miscprj/s_context.hc)
- aspcomplete
- virtuozzo (http://www.sw-soft.com/products/virtuozzo/)
- ensim (http://www.ensim.com/products/privateservers/index.html)
- Virtual Dedicated Server (http://www.sphera.com/prod_hostingdirector.php#sd)
- wmvare
- www.linux-vserver.org, www.linux-vserver.com
- VServer Patches and Patch Sets (http://www.13thfloor.at/VServer/Patches.shtml) — *Unofficial CTX Patches*
- VServer Wiki (http://vserver.strahlungsfrei.de/tiki-index.php)
- Free Virtual Private Server Solution (http://www.freevps.com/)
- virtuozzo (http://www.sw-soft.com/products/virtuozzo/)
- ensim (http://www.ensim.com/products/privateservers/index.html)
- aspcomplete

ToDo

1. Pořídit seznam software který tuto úlohu řeší.
2. Prozkoumat emulátory.
3. Popsat zběžně Free VPS.

FIXME: Virtuální privátní servery jsou jak již název napovídá servery běžící na jednom počítači. Jsou to samosatné server jenž mají oddělené systémy souborů (pokud se je nerozhodneme částečně sdílet) a oddělené procesy. Tzn. root v jednom serveru nemůže ovliňovat server jiný.

8.1. VirtualBox

* *Posponed*

Odkazy:

- VirtualBox.org (http://www.virtualbox.org/)
- debianWiki VirtualBox (http://wiki.debian.org/VirtualBox)
-
-

VirtualBox je virtualizační řešení od firmy Innotek které v tuto chvíli vlastní firma Sun. K dispozici je v několika různých variantách:

- VirtualBox OSE — pod licencí GNU
- VirtualBox PUEL — free pro nekomerční užití
- VirtualBox — komerční licence

Problémy instalace a konfigurace v Debianu jsou popsány na Debian Wiki (http://wiki.debian.org/VirtualBox).

Ahoj,

neodpovídáš na jabberu ;) tak píšu.

!!! ZVÍTĚZIL JSEM !!!

Ještě zkoumám další možnosti, ale v zásadě mi funguje komunikace mezi hostovaným počítačem (guest) a serverem (host).
Nastavení je následující:
Na hostovaném počítači nastavíme typ síťové karty "Hostitelské rozhraní" to je česky v anglickém rozhraní se to jemnuje "Host interface". V anglickém manuálu je to asi "Host-only networking". Pikantní na tom je, že tahle volba byla do VirtualBoxu podle manuálu přidána až ve verzi 2.2 a já mám verzi 1.6.6-dfsg-3. Sem z toho jelen. Navíc v menu vůbec nemám slovo o nějakém bridgování.
Takže zpět k nastavení hosta. Potom co nastavíš typ síťového rozhraní, zapíšeš do "Název rozhraní" anglicky "Interface Name" hodnotu třeba tap0, tu jsem použil já.
To je na straně hosta (guest) (virtuálního počítače všechno).

Na straně Serveru (host) je třeba se připojit k sítovému rozhraní virtuálního počítače. Musíš mít nainstalován balíček uml-utilities a pak provedeš:
tunctl -t tap0 -u radek # radek je jméno uživatele pod kterým běží virtualbox, tap0 je rozhraní stejné jméno jaké je nastaveno v konfgiguraci síťovky virtuánlího počítače.
ip link set up dev tap0 # Zapneme tap0
ifconfig tap0 192.168.51.41 netmask 255.255.255.0 # nastavíme tap0 síťovou adresu.
to je pro komunikaci mezi serverem a virtuálem všechno. Jo. Systém ve virtuálu si musí na eth0 nastavit odpovídající konfiguraci sítě aby mohl komunikovat. já jsem ve virtuálu udělal.

ifconfig eth0 192.168.51.42 netmask 255.255.255.0
A to je úplně všechno. pingá to, připojím se ssh.

Jako bonbónek můžu pustit virtuální počítač do internetu. To na virtuálu nadefinuju
ip route add default 192.168.51.41
A ještě musím nadefinovat nějaký nameserver. do souboru /etc/resolv.conf. U mě je to moje domácí brána
nameserver 172.31.1.1

Na straně serveru zapnu maškarádu a síťování.
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
A virtuál spokojeně aktualizuje balíčky z internetu.

To je pro dnešek vše.

- -
Radek

Některé problémy a jejich řešení

Pokud mi spadne při spouštění virtuálního stroje s chybou:

> Failed to open '/dev/net/tun' for read/write access. Please check the permissions of that node. Either run 'chmod 0666 /
> self a member of that group. Make sure that these changes are permanent, especially if you are using udev.
> VBox status code: -3100 (VERR_HOSTIF_INIT_FAILED).

Je problém v přístupu na /dev/net/tun. Jsem sice členem skupiny uml-net a zařízení patří také do této skupiny ale něco je špatného v kontrole oprávnění. V tomto případě se porovnává jen primární skupina. Pokud si před spuštěním VirtualBoxu nastavím správně primární skupinu, chyba se neobjeví.

```
$ newgrp uml-net
```

O chybě se dá dočíst v ticketu #1797 (http://www.virtualbox.org/ticket/1797).

8.1.1.

*

8.2. VServer

* *section id="vserver" xreflabel="VServer"*
* *print="**psselect -p 378-404 unix.ps |foldprn -s 32**"*
* *print="**psselect -p 382-403 unix.ps |foldprn -s 24**"*

Odkazy a zdroje:

- Linux-VServer Project (http://www.13thfloor.at/vserver/project) — záplaty
- Linux-VServer (http://www.linux-vserver.org/) — Wiki
- Virtual private servers and security contexts (http://www.solucorp.qc.ca/miscprj/s_context.hc)
- Virtual Dedicated Server (http://www.sphera.com/prod_hostingdirector.php#sd)
- http://www.linux-vserver.org, http://www.linux-vserver.com
- VServer Patches and Patch Sets (http://www.13thfloor.at/VServer/Patches.shtml) — *Unofficial CTX Patches*
- VServer Wiki (http://vserver.strahlungsfrei.de/tiki-index.php)
- VServer IP Setup (A Journey with Bash) (http://vserver.13thfloor.at/Stuff/VServer-IP-Setup-0.1.txt)
- Laurent Vallar - aka Val - corner (http://vallar.linuxfr.org/) — balíčky pro Debian
- util-vserver (http://savannah.nongnu.org/projects/util-vserver/) na Savannah nongnu (http://savannah.nongnu.org/) — The util-vserver project provides tools for kernels with the security context patch.
- . ()
- . ()

ToDo

1. Popsat historii VServeru.
2. Popsat nástroje pro práci s VServery. Popsat příkazy.

VServer je software pro realizaci privátních virtuálních serverů na linuxu. Všechny servery sdílejí společné jádro v němž je implementována technologie VServer. Nad tímto jádrem pak běží jako samostatné počítače s určitými omezeními. VServer je řešen jako záplata na jádro která implementuje potřebné služby a vytváří v jádře oddělené kontexty pro jednotlivé servery. Je k dispozici pro jádra řady 2.4 a jádra řady 2.6. Nevím, a mělo by se ověřit, jestli byl portován i na jádra řady 2.2.

Jednotlivé servery tedy sdílejí společný hardware. Mají k němu tudíž jen omezený přístup, prakticky zprostředkovaný přes hostiltelský server, server všech serverů. Každý ze serverů má vlastního roota, a tento nemá žádné možnosti ani práva ovlivňovat dení v jiných virtuálních serverech ani serveru hostitelském.

VServer je vhodný k nasazení pro hosting, kdy každý zákazník dostane vlastní virtuální server a šetříme místem, výkonem i materiálem.

`ctx` *Security Context*

8.2.1. Princip fungování (*Theory of operation*)

* ***FIXME:*** *dopsat*

vserver, jak již bylo řečeno sestáva ze záplaty jádra která modifikuje api o některá stávající volání a přidá nová.

`chroot()`

Omezení souborového systému, nastavení kořene souborového systému.

`chbind()`

Omezení IP prostoru.

`chcontext()`

Omezení prostoru procesů. Přepnutí do daného kontextu.

`drop_caps()`

Zahození (vzdání se) možností (capabilities)

8.2.2. Překlad a instalace

Jak jsem se již dříve zmínil, VServer je záplata na jádro a nástroje využívající nově přidané API. Je třeba si tedy opatřit přeložené jadro jenž tuto záplatu obsahuje, nebo si takové připravit.

Instalace sestává ze dvou částí:

- záplaty do jádra (záplaty ctx nebo vs)
- nástroje pro práci s virtuálními servery (balíčky vserver nebo util-vserver)

VServer je ve stádiu vývoje a doznal řady změn.

V současné době k 2003-07-02 používám na serverech Debian GNU/Linux Woody. K dispozici jsou tyto balíčky:

- `kernel-patch-ctx` `17-1` v `testing`
- `vserver` `0.22-11` v `testing`

Nejprve je třeba ozáplatovat jádro. Nainstalujeme si tedy záplatu z `sarge`, protože ve `woody` není.

```
moon# apt-get install -t sarge kernel-patch-ctx
```

a můžeme jádro přeložit. Při překladu postupujeme podle Překlad jádra se záplatami

```
moon# make-kpkg clean
 moon# make-kpkg --append_to_version -moon --revision 1 \
 --added-patches ctx \
 kernel_image modules_image
```

Po jádru si přeložím `vserver`

```
# apt-get source vserver
 # cd vserver-0.22
 # dpkg-buildpackage -b -uc
```

Přeložené jádro nainstalujeme, překopírujeme `deb` soubor do adresáře s lokálními balíčky `/root/debs` a aktualizujeme skriptem `/root/bin/update-debs`. Poté provedem vlastní instalaci a pro jistotu jen na disketu.

```
# apt-get install kernel-image-2.4.20-moon
```

Aby bylo možno do virtuálního serveru instalovat i jinou distribuci než právě stabilní (woody) bylo třeba vykopírovat z nejaktuálnějšího balíčku `debootstrap_0.1.17.31_i386.deb` adresář ze skripty a ty uložit do `/usr/lib/debootstrap/scripts`

Dále uvádím příklady záplatování překladů různých jader pro konkrétní počítače tak jak jsem je použil.

8.2.2.1. Překlad VServeru na jádro 2.4.24

Po změnách na jednom serveru (`moon`) jsem se rozhodl upgradovat taky jádro. O tom jak jsem postupoval, je tato sekce. Všechny verze jsou aktuální ke dni 2004-02-18.

Jádro 2.4.24 jsem získal, spolu s potřebnými nástroji z Backports (backports.org). Pro úplnost uvádím související informace. V souboru `/etc/apt/sources.list` mám řádek:

```
deb http://www.backports.org/debian woody kernel-source-2.4.24 kernel-package
```

a s pomocí **apt-get** nainstaluji zdroje jádra a pomocné nástroje

```
# apt-get update
# apt-get install kernel-source-2.4.24 kernel-package
```

Stáhl jsem si patch na jádro 2.4.24 do adresáře `/root/tmp`

```
# wget http://www.13thfloor.at/vserver/s_release/v1.26/patch-2.4.24-vs1.26.diff.gz
```

Zdroje jádra je třeba rozbalit a vytvořit symbolický odkaz

```
# cd /usr/src
# tar xjf kernel-source-2.4.24.tar.bz2
# ln -s kernel-source-2.4.24 linux
```

Do zdrojů jádra si ještě nakopíruji konfiguraci běžícího jádra 2.4.20

```
# cp /boot/config-2.4.20-moon linux/moon.0
```

Připravené zdroje jádra je třeba zazáplatovat

```
# cd linux
# zcat /root/tmp/patch-2.4.24-vs1.26.diff.gz| patch -p1
```

v průběhu záplatování se vyskytl problém:

```
patching file include/net/ip.h
```

```
Hunk #1 FAILED at 29.
1 out of 1 hunk FAILED -- saving rejects to file include/net/ip.h.rej
patching file include/net/route.h
Hunk #1 succeeded at 33 (offset 1 line).
Hunk #2 FAILED at 168.
1 out of 2 hunks FAILED -- saving rejects to file include/net/route.h.rej
...
Hpatching file net/ipv4/udp.c
Hunk #1 succeeded at 116 (offset 6 lines).
Hunk #2 succeeded at 176 (offset 6 lines).
Hunk #3 succeeded at 219 (offset 6 lines).
Hunk #4 succeeded at 251 (offset 6 lines).
Hunk #5 succeeded at 309 (offset 6 lines).
Hunk #6 FAILED at 557.
Hunk #7 succeeded at 1312 (offset 262 lines).
1 out of 7 hunks FAILED -- saving rejects to file net/ipv4/udp.c.rej
```

tento je třeba řešit ručne. Vypadá to že oprava je netriviální. Tím jsme skončili a samotný překlad jenž bychom provedi následujícím způsobem nemá smysl.

Ovšem řešení se našlo v použití jiného patche

```
# cd /root/tmp
# wget http://vserver.13thfloor.at/Experimental/patch-2.4.24-1-vs1.24.1.diff
# cd /usr/src/linux
# cat /root/tmp/patch-2.4.24-1-vs1.24.1.diff| patch -p1
```

Tato záplata proběhla bez jakýchkoliv problémů. Pro pořádek jsem si přejmenoval adresář se zdroji jádra

```
# mv kernel-source-2.4.24 kernel-source-2.4.24-vs1.24.1
```

Poté jsem již mohl přistoupit k překladu

```
# cd /usr/src
# ln -sf kernel/source-2.4.24-vs1.24.1 linux
# cd /usr/src/linux
# make-kpkg clean
# make-kpkg --append-to-version -moon --revision 1 \
            --config menu \
            kernel_image modules_image
```

nahrávám konfiguraci z aktuálního jádra uloženou v `moon.0`, ukládám konfiguraci do `moon.1` beze změn a spouštím překlad opuštěním konfiguračního menu s uložením konfigurace.

K takto připravenému jádru si připravíme nastroje.

```
deb-src http://debian:9999/main sarge main contrib non-free
```

```
# apt-get source --build vserver
```

Je potřeba nechat v `/etc/apt/sources.list` odkazy jen na stabilní větev a naše lokální balíčky.

```
# apt-get install --reinstall vserver
```

8.2.2.2. Překlad VServeru pro jádro 2.4.25 z backports.org

Ne zcela dokonalý a regulérní postup:

```
# cd /root/debs
# wget http://vallar.linuxfr.org/debian/kernel-patch-vs_0.3-1/kernel-patch-vs_0.3-1_all.deb
# update-debs
# apt-get update
# apt-get install kernel-patch-vs
# ~/sbin/build-kernel yoda 5
```

Kde skript pro sestavení jádra `build-kernel` vypadá takto:

```
#!/bin/sh
export PATCH_THE_KERNEL=YES
cd /usr/src
rm -fr modules
tar xjf alsa-driver.tar.bz2
tar xzf cipe.tar.gz
tar xzf thinkpad.tar.gz
tar xzf pcmcia-cs.tar.gz
cd linux
make-kpkg clean
make-kpkg --added-patches vs --append-to-version -$1 --revision $2 \
          --config menu kernel_image modules_image
```

V menu se objevila nová volba `Block devices/Virtual Root device support`

Po překladu instalujeme nové jádro:

```
# mv /usr/src/*.deb /root/debs
# update-debs
# apt-get update
# apt-get upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
  irate-client-motif
The following packages will be upgraded
  alsa-modules-2.4.25-yoda cipe-2.4.25-yoda kernel-image-2.4.25-yoda thinkpad-modules-2.4.25-yoda
4 packages upgraded, 0 newly installed, 0 to remove and 1  not upgraded.
1 packages not fully installed or removed.
Need to get 0B/3152kB of archives. After unpacking 20.5kB will be used.
Reading changelogs...
```

Do souboru `/etc/apt/sources.list` přidáme řádku

```
deb http://www.backports.org/debian woody vserver
```

a doinstalujeme nástoje

```
# apt-get install vserver
```

8.2.2.3. Překlad VServeru vs1.27 na jádro 2.4.25 z backports.org

* *section id="vs1.27-linux2.4.25-manual-patch"*
* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Následující příklad je spíše než ukázkou překladu jádra ukázkou řešení problému při záplatování. Vlastně jsem prováděl celé záplatování zbytečně neb jsem vycházel z mylného předpokladu že nový patch vyřeší můj „problém“. Problém s nefungováním příkazu **hostname** pro nastavení jména počítače ve virtuálním serveru. Jak

jsem posléze zjistil toto se nastavuje v hostitelském serveru v konfiguračním souboru virtuálního stroje. Dále tedy následuje uvedený, „zbytečný“ postup.

Protože jsem nebyl spokojený s předchozím jádrem, rozhodl jsem se připravit si jádro s poslední stabilní záplatou vserveru vs1.27. Postupoval jsem následovně.

Stáhl jsem si poslední záplatu a utility.

```
# wget http://www.13thfloor.at/vserver/s_release/v1.27/patch-2.4.25-vs1.27.diff.bz2
# wget http://www.13thfloor.at/vserver/s_release/v1.27/util-vserver-0.29.3.tar.bz2
```

Záplaty jsem aplikoval na zdroje jádra.

```
# cd /usr/src/linux
# bzcat /home/radek/work/vserver/patch-2.4.25-vs1.27.diff.bz2 \
      | patch -p1 | tee patch.log
```

Celý průběh záplatování jsem si nechal zapsat do souboru `patch.log` aby jej mohl podrobně prozkoumat. Záplatování s v několika kouscích (*hunk*) nepovedlo, tak se podíváme na neúspěchy.

```
# grep FAILED patch.log
Hunk #1 FAILED at 807.
Hunk #2 FAILED at 1088.
2 out of 2 hunks FAILED -- saving rejects to file arch/ppc64/kernel/misc.S.rej
Hunk #1 FAILED at 29.
1 out of 1 hunk FAILED -- saving rejects to file include/net/ip.h.rej
Hunk #2 FAILED at 168.
1 out of 2 hunks FAILED -- saving rejects to file include/net/route.h.rej
Hunk #6 FAILED at 557.
1 out of 7 hunks FAILED -- saving rejects to file net/ipv4/udp.c.rej
```

Nezbylo mi než se zahloubat do kódu a aplikovat neúspěšné kousky ručně.

Začal jsem odmítnutým kódem v souboru `include/net/ip.h.rej`. Tato část je nejjednodušší.

```
***************
*** 29,34 ****
  #include <linux/netdevice.h>
  #include <linux/inetdevice.h>
  #include <linux/in_route.h>
  #include <net/route.h>
  #include <net/arp.h>

--- 29,35 ----
  #include <linux/netdevice.h>
  #include <linux/inetdevice.h>
  #include <linux/in_route.h>
+ #include <linux/vcontext.h>
  #include <net/route.h>
  #include <net/arp.h>
```

Tady byla oprava triviální. Za řádku

```
#include <linux/in_route.h>
```

jsem vsunul potřebnou řádku

```
#include <linux/vcontext.h>
```

Poté jsem pokračoval souborem `include/net/route.h.rej`. Tady je neúspěšný kód již delší.

```
***************
*** 167,172 ****
  static inline int ip_route_connect(struct rtable **rp, u32 dst, u32 src, u32 tos, int oif)
  {
        int err;
        err = ip_route_output(rp, dst, src, tos, oif);
        if (err || (dst && src))
                return err;
--- 168,211 ----
  static inline int ip_route_connect(struct rtable **rp, u32 dst, u32 src, u32 tos, int oif)
  {
        int err;
+       struct iproot_info *ip_info = current->ip_info;
+       if (ip_info != NULL) {
+               __u32 ipv4root = ip_info->ipv4[0];
+               if (ipv4root != 0) {
+                       int n = ip_info->nbipv4;
+                       if (src == 0) {
+                               if (n > 1) {
+                                       u32 foundsrc;
+                                       int i;
+                                       err = ip_route_output(rp, dst, src, tos, oif);
+                                       if (err) return err;
+                                       foundsrc = (*rp)->rt_src;
+                                       ip_rt_put(*rp);
+                                       for (i=0; i<n; i++){
+                                               u32 mask = ip_info->mask[i];
+                                               u32 ipv4 = ip_info->ipv4[i];
+                                               u32 netipv4 = ipv4 & mask;
+                                               if ((foundsrc & mask) == netipv4) {
+                                                       src = ipv4;
+                                                       break;
+                                               }
+                                       }
+                               }
+                               if (src == 0)
+                                       src = dst == 0x0100007f
+                                               ? 0x0100007f: ipv4root;
+                       } else {
+                               int i;
+                               for (i=0; i<n; i++) {
+                                       if (ip_info->ipv4[i] == src) break;
+                               }
+                               if (i == n)
+                                       return -EPERM;
+                       }
+                       if (dst == 0x0100007f && !vx_check(0, VX_ADMIN))
+                               dst = ipv4root;
+               }
+       }
        err = ip_route_output(rp, dst, src, tos, oif);
        if (err || (dst && src))
                return err;
```

Po ruční opravě vypadá rozdíl takto

```
@@ -156,6 +157,45 @@
                                        .dport = dport } } };

        int err;
+       struct iproot_info *ip_info = current->ip_info;
+       if (ip_info != NULL) {
+               __u32 ipv4root = ip_info->ipv4[0];
+               if (ipv4root != 0) {
+                       int n = ip_info->nbipv4;
+                       if (src == 0) {
+                               if (n > 1) {
+                                       u32 foundsrc;
+                                       int i;
+                                       err = ip_route_output_flow(rp, &fl, sk, 0);
+                                       if (err) return err;
+                                       foundsrc = (*rp)->rt_src;
+                                       ip_rt_put(*rp);
```

```
+                                       for (i=0; i<n; i++){
+                                               u32 mask = ip_info->mask[i];
+                                               u32 ipv4 = ip_info->ipv4[i];
+                                               u32 netipv4 = ipv4 & mask;
+                                               if ((foundsrc & mask) == netipv4) {
+                                                       src = ipv4;
+                                                       break;
+                                               }
+                                       }
+                               }
+                               if (src == 0)
+                                       src = dst == 0x0100007f
+                                               ? 0x0100007f: ipv4root;
+                       } else {
+                               int i;
+                               for (i=0; i<n; i++) {
+                                       if (ip_info->ipv4[i] == src) break;
+                               }
+                               if (i == n)
+                                       return -EPERM;
+                       }
+                       if (dst == 0x0100007f && !vx_check(0, VX_ADMIN))
+                               dst = ipv4root;
+               }
+       }
+
```

Poslední kousek který neuspěl je `include/net/ipv4/udp.c.rej`

```
***************
*** 540,545 ****
                rt = (struct rtable*)sk_dst_check(sk, 0);

        if (rt == NULL) {
                err = ip_route_output(&rt, daddr, ufh.saddr, tos, ipc.oif);
                if (err)
                        goto out;
--- 557,574 ----
                rt = (struct rtable*)sk_dst_check(sk, 0);

        if (rt == NULL) {
+               struct iproot_info *ip_info = current->ip_info;
+
+               if (ip_info != NULL) {
+                       __u32 ipv4root = ip_info->ipv4[0];
+                       if (ipv4root) {
+                               if (daddr == 0x0100007f &&
+                                       !vx_check(0, VX_ADMIN))
+                                       daddr = ipv4root;
+                               if (ufh.saddr == 0)
+                                       ufh.saddr = ipv4root;
+                       }
+               }
                err = ip_route_output(&rt, daddr, ufh.saddr, tos, ipc.oif);
                if (err)
                        goto out;
```

Záplatu jsem vsunul za řádku 618

```
.dport = dport } } };

@@ -599,6 +616,19 @@
```

```
                                .uli_u = { .ports =
                                          { .sport = sk->sport,
                                            .dport = dport } } };
+               struct iproot_info *ip_info = current->ip_info;
+
+               if (ip_info != NULL) {
+                       __u32 ipv4root = ip_info->ipv4[0];
+                       if (ipv4root) {
+                               if (daddr == 0x0100007f &&
+                                       !vx_check(0, VX_ADMIN))
+                                       daddr = ipv4root;
+                               if (fl.nl_u.ip4_u.saddr == 0)
+                                       fl.nl_u.ip4_u.saddr = ipv4root;
+                       }
+               }
+
                err = ip_route_output_flow(&rt, &fl, sk, !(msg->msg_flags&MSG_DONTWAIT));
                if (err)
                        goto out;
```

Záplatováním souboru `arch/ppc64/kernel/misc.S` jsem se nazabýval, neb jádro použiji jen na platformě Intel.

Uschování ozáplatovaných zdrojů

```
# cd /usr/src
# mv kernel-source-2.4.25 kernel-source-2.4.25-vs1.27
# cd kernel-source-2.4.25-vs1.27
# make mrproper
# cd ..
# tar cf kernel-source-2.4.25-vs1.27.tar kernel-source-2.4.25-vs1.27
# bzip2 --best kernel-source-2.4.25-vs1.27.tar.bz2
```

Vlastní překlad jádra

```
# /root/sbin/build-kernel yoda 6
```

8.2.2.4. Instalace nástrojů pro vserver

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Ke starším verzím VServeru se používal balíček nástrojů stejného jména. V Debian Sarge jsou k dispozici dva balíčky: util-vserver a vserver-debiantools. Jejich instalace je tím pádem snadná:

```
# aptitude install util-vserver vserver-debiantools
```

V Debian Woody to bylo trochu obtížnější, používal jsem k tomuto účelu balíčky z Debian Backports (Backports.ORG) (http://www.backports.org/). Postup instalace byl následující:

1. Nejdříve bylo nutné upravit `/etc/apt/sources.list`. Podle toho jestli jsem přistupoval přímo, nebo používal apt-proxy jsem do uvedeného souboru vložil jednu z následujících řádek:

   ```
   deb http://www.backports.org/debian/ woody util-vserver
   ```

   nebo

```
deb http://debian:9999/backports woody util-vserver
```

První řádka přistupuje k balíčkům na Debian Backports (Backports.ORG) (http://www.backports.org/) přímo, druhá používá moji lokální apt-proxy.

2. Po úpravě konfiguračního souboru provedeme aktualizaci databáze dostupných balíčků:

```
# apt-get update
```

3. A nyní můžem provést samotnou instalaci:

```
# apt-get install util-vserver
```

8.2.2.5. Instalace na Debian Sarge s použitím jádra 2.4.27

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Na Sarge je instalace jednodušší než na Woody. Nástroje i patch máme k dispozici prímo v distribuci. Překladu vlastního jádra se ovšem nevyhneme. K překladu budeme potřebovat záplatu (patch) jádra kterou si nainstalujeme z balíku `kernel-patch-vserver`.

```
# aptitude install kernel-patch-vserver
```

Vlastní překlad provedeme posloupností příkazů

```
# cd /usr/src
# tar xjf kernel-source-2.4.27
# rm linux
# ln -s kernel-source-2.4.27 linux
# export PATCH_THE_KERNEL=YES
# make-kpkg clean
# make-kpkg --append-to-version -vs --revision 1 \
#           --added-patches vs \
#           --config menu kernel_image modules_image
```

A doinstalujeme potřebné nástroje.

```
# aptitude install util-vserver vserver-debiantools
```

Pro instalaci virtuálních serverů většinou používám jednoduchý skript. Při tom vycházím z následující krátké šablony, kterou si zkopíruji a upravím. Skript před započetím instalace odstraní data a konfigurace předchozí instalace, jestli toto existuje.

```
#!/bin/sh
# Rebuild virtual server

NAME=srv1
IP=312.299.1.262

vserver $NAME stop
rm -r /home/vservers/$NAME /etc/vservers/$NAME.conf

newvserver --hostname $NAME --domain example.org --ip $IP
```

8.2.3. Základní konfigurace

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Konfigurace VServerů je uložena v souboru `/etc/vservers.conf` a v adresáři `/etc/vservers`. V uvedeném adresáři se nachází konfigurační soubor `newvservers-vars` z balíčku vserver-debiantools, který popisuje jakým způsobem bude probíhat instalace nového virtuálního server příkazem **newvserver**. Dále jsou/budou zde konfigurační soubory či adresáře pro jednotlivé virtuální servery. Mimo to jsou zde dva adresáře `.defaults` a `.distributions`.

V adresáři `.defaults` se nachází mimo jiné adresář `.vdirbase` jenž je symbolickým odkazem na kořenový adresář ve kterém jsou root svazky jednotlivých virtuálních serverů.

Konfigurace vserveru je v souboru `/etc/vservers.conf` a adresáři `/etc/vservers`

Soubor `/etc/vservers/util-vserver-vars`

```
PKGLIBDIR='/usr/lib/util-vserver'
SBINDIR='/usr/sbin'
VROOTDIR='/var/lib/vservers'
```

8.2.4. Balíček programů `util-vserver`

* *section id="util-vserver"*
* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

Startovací skripty v `/etc/init.d`: `rebootmgr`, `vprocunhide`, `vservers-default`, `vservers-legacy`.

/usr/sbin/chbind

FIXME:

/usr/sbin/exec-cd

FIXME:

/usr/sbin/lsxid

FIXME:

/usr/sbin/chxid

FIXME:

/usr/sbin/vps

FIXME:

/usr/sbin/showattr

FIXME:

/usr/sbin/setattr

FIXME:

/usr/sbin/reducecap

FIXME:

/usr/sbin/vdu

FIXME:

/usr/sbin/vattribute

FIXME:

/usr/sbin/vcontext

FIXME:

/usr/sbin/vlimit

FIXME:

/usr/sbin/vkill

FIXME:

/usr/sbin/vnamespace

FIXME:

/usr/sbin/vrsetup

FIXME:

/usr/sbin/vsched

FIXME:

/usr/sbin/vserver-stat

FIXME:

/usr/sbin/vserver-info

FIXME:

/usr/sbin/vuname

FIXME:

/usr/sbin/vfiles

FIXME:

/usr/sbin/chcontext

FIXME:

/usr/sbin/vapt-get

FIXME:

/usr/sbin/vpstree

FIXME:

/usr/sbin/vrpm

FIXME:

/usr/sbin/vserver

FIXME:

/usr/sbin/vserver-copy

FIXME:

/usr/sbin/vsomething

FIXME:

/usr/sbin/vtop

FIXME:

/usr/sbin/vyum

FIXME:

/sbin/vshelper

FIXME: ?? emit command in virtual machine ??

V balíčku jsou programy:

- **chbind**
- **chcontext**
- **rebootmgr**
- **reducecap**
- **vlimit**
- **vdu**
- **vfiles**
- **vkill**
- **vserver-stat**
- **vpstree**
- **vrpm**
- **vserver**
- **vserver-copy**
- **vtop**
- **vps**

8.2.5. Nástroje `vserver-debiantools`

* *section id="vserver-debiantools" xreflabel="vserver-debiantools"*

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

/usr/sbin/dupvserver

FIXME:

/usr/sbin/newnfsvserver

FIXME:

/usr/sbin/newvserver

FIXME: Vytvoření nového virtuálního stroje založeného na distribuci Debian

```
# newvserver --hostname help --domain mydomain.cz --ip 1.2.3.12
```

/usr/sbin/stripserver

FIXME:

`/etc/vservers/newvservers-vars`

FIXME:

Dalším balíčkem který by se nám mohl na Debianu hodit je `vserver-debiantools`. Tento obsahuje několik skriptů pro práci s virtuálními počítači. Jediné verze které jsou dostupné jsou z Debian Sarge. Balíček neobsahuje žádno spustitelné binární soubory jen skripty. Proto jsem se rozhodl si ho portovat do Woody. Na překládajícím stroji tedy nastavím v `/etc/apt/sources.list` přístup ke zdrojům:

```
deb-src http://localhost:9999/main sarge main
```

Zdroje stáhnu a rovnou přeložím.

```
# apt-get source -b vserver-debiantools
```

Vše proběhlo bez problémů. Nakonec jsem ovšem provedl pár úprav v skriptech které zohledňují moje potřeby a vytvořil novou verzi balíčku. tu dále distribuuji a používám na svých serverech.

* ***FIXME:****Vystavit tuto verzi na internetu.*

Konfigurace se nachází v souboru `/etc/vservers/newvserver-vars` který je shell skriptem jenž se načítá v průběhu práce. Nastavíme jej dle potřeby. Ukázková instalace používající apt-proxy.

```
# Configuration file for newvserver
# See man newvserver for the variables that you can set here.
DIST="woody"
MIRROR="http://debian:9999/main"
APTPROXY="debian:9999"
SOURCESLIST="
# RootStuff
#deb file:/root/stuff/pkg/deb woody kernel-dell kernel-ird vserver

deb http://$APTPROXY/main woody main contrib
deb http://$APTPROXY/non-US woody/non-US main contrib
deb http://$APTPROXY/security woody/updates main
"
```

Ukázka vytvoření virtuálního serveru `grpw`:

```
# newvserver --
```

8.2.6. Vytvoření nového virtuálního serveru

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

V této částu uvádím příklady instalací virtuálních serverů.

vserver již máme úspěšně nainstalován a tak přikročíme k vytvoření virtuálního serveru. Pro tento účel se používá skript **newvserver**

```
# export LANG="C"
# newvserver --hostname money --domain moraviapress.cz \
             --ip 10.16.66.25 --dist sarge
```

Znovuvytvoření serveru `haiku`. Do konfiguračního souboru zapíšeme `/etc/vserver.conf`:

```
MIRROR=http://debian:9999/main
```

Tím zajistíme že hlavní část balíčků se bude stahovat z naší debian proxy cache. Následujícími příkazy znovu vytvoříme virtuální server

```
# vserver haiku stop
# rm -fr /var/lib/vserver/haiku
# rm /etc/vservers/haiku.conf
# LANG=C newvserver --hostname haiku --domain moraviapress.cz \
                    --ip 10.16.66.131 --dist woody
```

Ručně se virtuální server vytvoří příkazy

```
# mkdir /vservers/XX
# cd /vservers/XX
# cp -ax /sbin /bin /etc /usr /var /dev /lib .
# mkdir proc tmp home
# chmod 1777 tmp
```

Takovýto postup je ovšem velmi neštastný.

8.2.6.1. Příklad vytvoření virtuálního serveru na `oritu`

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Nainstalujeme si potřebné nástroje

```
# apt-get install debootstrap
```

Vytvoříme prostor pro server. Na `/var` mám málo místa a tak vytvořím prostor na `/home` a vytvořím symbolický odkaz.

```
# mkdir /home/hrad
# cd /
# ln -s /home/hrad /var/lib/vservers/
```

Poté zatáhnu základ systému

```
# debootstrap --arch i386 woody \
              /var/lib/vservers/hrad http://www.debian.cz/debian
```

8.2.6.2. Příklad vytvoření virtuálního serveru ray na počítači yoda

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Pro instalaci použijeme stejně jako v předchozím případu **debootstrap** Vytvoříme prostor pro server.

```
# mkdir /var/lib/vservers/ray
```

Poté zatáhnu základ systému

```
# debootstrap --arch i386 woody \
              /var/lib/vservers/ray http://www.debian.cz/debian
```

8.2.6.3. Ukázka vytvoření virtuálního serveru tree

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

S použitím skriptu **debian-newserver.sh** který jseem získal z webu No War (http://www.paul.sladen.org/) Paula Sladena

```
# wget http://www.paul.sladen.org/vserver/debian/debian-newvserver.sh
```

```
# export LANG=C
# ~/sbin/debian-newvserver.sh --hostname tree --domain hnilica.cz \
                              --ip 192.168.172.2 --dist woody
```

Odpověděl jsem mu na pár otázek:

Time Zone Configuration

Europe / Prague

Password setup

Shall I enable md5 passwords? Yes

Shall I enable shadow passwords? Yes

Enter a password for root:

Shall I create a normal user account now? No

Select tasks to install

Nevybral jsem žádný task.

eximconf

Vybral jsem volbu 5 (žádná konfigurace)

V adresáři `/etc/vservers` přibyla definice serveru `tree.conf`

```
S_HOSTNAME="tree"
IPROOT="192.168.172.2"
IPROOTDEV="eth0"
```

```
ONBOOT="yes"
S_NICE=""
S_FLAGS="lock nproc"
ULIMIT="-H -u 256 -n 1024"
S_CAPS="CAP_NET_RAW"

# *NOT* DNS domain name, for NIS only
S_DOMAINNAME=""
```

Do takto vytvořeného virtuálního serveru jsem vstoupil

```
# vserver tree enter
```

a můžeme pracovat.

8.2.6.4. Vytvoření virtuálního serveru `kiki` na počítači `oritu`

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Nejdříve si stáhneme skript pro vytváření virtuálních serverů

```
# cd ~/bin
# wget http://www.paul.sladen.org/vserver/debian/debian-newvserver.sh
# chmod u+x debian-newvserver.sh
```

a upravíme

```
--- debian-newvserver.sh.orig   Sat Apr  3 19:15:52 2004
+++ debian-newvserver.sh        Sat Apr  3 19:27:40 2004
@@ -10,7 +10,7 @@
 # Configurable items:

 # Root directory of your virtual servers (probably shouldn't change this)
-VSERVER_ROOT="/vservers"
+VSERVER_ROOT="/var/lib/vservers"

 # Packages to install in addition to the base defaults
 # MUST INCLUDE ALL DEPENDENCIES (seperated by "," commas)
@@ -28,13 +28,13 @@

 # Architecture: overide on non-Debian host such as Redhat
 # otherwise dpkg will detect whether we are i386/powerpc/sparc/etc
-ARCH=""
+ARCH="i386"

 # Which debian distribution (warning: this has only been tested with woody)
 DIST="woody"

 # Local or nearest location of a debian mirror (must include the '/debian')
-MIRROR="http://ftp.uk.debian.org/debian"
+MIRROR="http://www.debian.cz/debian"

 # Default network interface for vservers:
 INTERFACE="eth0"
```

Připravíme se adresář pro náš virtuální server

```
# mkdir /home/vserver/kiki
# ln -s /home/vserver/kiki /var/lib/vservers/
```

a můžeme instalovat

```
# export LANG=C
# bin/debian-newvserver.sh --hostname kiki --domain giga-net.cz \
                           --ip 213.29.4.55 --dist woody
```

8.2.6.5. Virtuální server hod2

* *section id="vserver.hod2" condition="author"*
* *svninfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $ $HeadURL$"*
* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

Instalace serveru hodonin
==========

omega:~# cat rebuild-hod2
#!/bin/sh

Rebuild the virtual server hod2 for domain hodonin.cz

NAME=hod2
IP=213.29.4.58

vserver $NAME stop
rm -r /home/vservers/$NAME /etc/vservers/$NAME.conf

newvserver --hostname $NAME --domain hodonin.cz --ip $IP
#EOF

omega:~# ./rebuild-hod2
omega:~# cp /etc/apt/apt.conf.d/10aptitude /etc/vservers/.defaults/vdirbase/hod/etc/apt/apt.conf.d/

omega:~# vserver hod start
omega:~# vserver hod enter

hod:~# aptitude update
hod:~# aptitude upgrade

hod:~# aptitude install less

hod:~# aptitude install ssmtp
Who gets mail for userids < 1000? admin@giga-net.cz
Name of your mailhub? smtp.giga-net.cz
What domain to masquerade as? hodonin.giga-net.cz
Allow override of From: line in email header? <YES>

```
hod:~# aptitude install postgresql-client

Alternativně apache nebo apache2

hod:~# aptitude install apache
The following packages are RECOMMENDED but will NOT be installed:
  debconf-utils file libsasl2-modules perl-doc
Enable su exec <No>
hod:/etc/apache# diff httpd.conf.orig httpd.conf
182c182
< #Listen 12.34.56.78:80
---
> Listen 213.29.4.12:80
190c190
< #BindAddress *
---
> BindAddress 213.29.4.12
260c260
< ServerAdmin webmaster@localhost
---
> ServerAdmin admin@giga-net.cz
273c273
< ServerName localhost
---
> #ServerName localhost
275c275
< #ServerName new.host.name
---
> ServerName hodonin.giga-net.cz
838c838
< AddDefaultCharset on
---
> AddDefaultCharset off
hod:/etc/apache#

hod:~# aptitude install libapache-mod-php4 php4

Ověřit že /etc/apache/modules.conf obsahuje řádku
LoadModule php4_module /usr/lib/apache/1.3/libphp4.so
--------------------


Using Apache2:
---------------
hod:~# aptitude install apache2
The following packages are RECOMMENDED but will NOT be installed:
  file libsasl2-modules

hod:/# aptitude install libapache2-mod-php4 php4
--------------

hod:~# aptitude install php4-pgsql
```

```
The following packages are RECOMMENDED but will NOT be installed:
  debconf-utils perl-doc

hod:/# aptitude install drupal
Automatically create Drupal database? <Yes>
Run database update script?  <No>
Database engine to be used with Drupal PostgreSQL
Database server for Drupal’s database isql.giga-net.cz

Drupal database name   drupal
Remove Drupal database when the package is removed? <No>
Remove former database backups when the package is removed? <No>
Web server(s) that should be configured automatically apache2

Poznámky:
---------
Vytvoření databáze
CREATE USER drupal WITH PASSWORD ’drupal4’
CREATE DATABASE drupal WITH OWNER=druapl TEMPLATE=template0 ENCODING=’LATIN2’

aptitude install apache
aptitude install libapache-mod-php4

echo "LoadModule php4_module /usr/lib/apache/1.3/libphp4.so" >> /etc/apache/modules.conf

postgres@isql:/usr/share/doc/postgresql$ createlang plpsql drupal
drupal=> ALTER TABLE sessions ALTER uid SET DEFAULT 0;

Konfigurace apache:
-------------------
Drupal v adresáři, třebas /usr/local/share/drupal4.6 publikujeme v konfiguraci takto:
 Alias /drupal4.6 /usr/local/share/drupal4.6
 <Directory /usr/local/share/drupal4.6/>
  Options +FollowSymLinks
  AllowOverride All
  order allow,deny
  allow from all
 </Directory>

soubor .htaccess v adresáři drupalu pak popisuje další konfiguraci
```

8.2.7. Instalace bind9 na virtuální server koky

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Bind9 z Debian Woody má problém, nelze ho přímo bez úprav spustit. Důvody osvětlí tato korespondendce a další dokumenty

- http://www.paul.sladen.org/vserver/archives/200211/0172.html
- Paul Sladen’s s_context/vserver FAQ (http://www.paul.sladen.org/vserver/faq/#bind9)

- http://www.paul.sladen.org/vserver/archives/200302/0186.html
- http://www.paul.sladen.org/vserver/archives/200302/0196.html
- http://www.paul.sladen.org/vserver/archives/200301/0034.html
- http://www.paul.sladen.org/vserver/archives/200302/0200.html
- http://www.solucorp.qc.ca/howto.hc?projet=vserver&id=72

```
koky:/# apt-get install bind9
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  libisccc0 libisccfg0
The following NEW packages will be installed:
  bind9 libisccc0 libisccfg0
0 packages upgraded, 3 newly installed, 0 to remove and 4  not upgraded.
Need to get 377kB of archives. After unpacking 913kB will be used.
```

8.2.8. Základní konfigurace virtuálního serveru

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Základní konfigurace je popsána v souboru `/etc/vservers/`*`jméno`*`.conf`. V tomto souboru definujeme řadu proměnných popisující virtuální server.

IPROOT

FIXME:

IPROOTDEV

FIXME:

IPROOTMASK

FIXME:

IPROOTBCAST

FIXME:

ONBOOT

FIXME:

S_CAPS

FIXME:

S_CONTEXT

FIXME:

S_DOMAINNAME

FIXME:

S_HOSTNAME

FIXME:

S_NICE

FIXME:

S_FLAGS

FIXME:

ULIMIT

FIXME:

8.2.9. Seznam programů a nástrojů

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

`/usr/sbin/chcontext`

mění bezpečnostní kontext / virtuální server

`/usr/sbin/chbind`

Uzamyká/svazuje proces a jeho potomky na specifickou IP adresu.

Příklad spouštěný jako root:

```
# httpd is running
netstat -atn | grep ":80 "
# We see a line like this
# tcp    0   0 0.0.0.0:80    0.0.0.0:*      LISTEN
# Now we restart httpd, but we lock it so it
# uses only the main IP of eth0
/usr/sbin/chbind --ip eth0 /etc/rc.d/init.d/httpd restart
netstat -atn | grep ":80 "
# Now we see a line like
# tcp    0   192.168.1.1:80    0.0.0.0:*      LISTEN
# httpd conf was not changed.
# Now restart it normally
/etc/rc.d/init.d/httpd.restart
# Now we experiment with client socket
# Log using telnet
telnet localhost
```

`/etc/sbin/reducecap`

Zmenšuje/redukuje možnosti procesu a jeho potomků. Ani setuid nebude schopen získat více možností (*capability*)

`/etc/sbin/newvserver`

vytváření nových virtuálních serverů. frontend pro vserver-admin

newvserver [`--hostname` *`hostname`*] [domain] [ip]

`/etc/sbin/vdu`

ořezaná verze **du** oznamující velikost použitého prostoru

`/etc/sbin/vkill`

FIXME:

`/etc/sbin/vps`

FIXME:

`/etc/sbin/vpstree`

FIXME:

8.2.9.1. Paul Sladen newvserver

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

- http://www.paul.sladen.org/vserver/archives/200211/0150.html
- debian-newvserver.sh (http://www.paul.sladen.org/vserver/debian/debian-newvserver.sh)

http://www.paul.sladen.org/vserver/debian/

8.2.10. Skript `vsmaker`

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Odkazy a zdroje:

- Virtual private servers and security contexts (http://tomshiro.org/vsmaker)

8.2.11. vserver

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Odkazy a zdroje:

- www.linux-vserver.org, www.linux-vserver.com
- VServer Patches and Patch Sets (http://www.13thfloor.at/VServer/Patches.shtml) — *Unofficial CTX Patches*
- VServer Wiki (http://vserver.strahlungsfrei.de/tiki-index.php)
- Paul Sladen's s_context/vserver FAQ (http://www.paul.sladen.org/vserver/faq)
- ... ()

ToDo

1. Pořídit seznam software který tuto úlohu řeší.

8.2.12. Postup prekladu

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Podle Installing a ctx kernel on Debian (http://vserver.strahlungsfrei.de/tiki-index.php?page=GetTheKernelAndInstall)

Before you start please install the package named "kernel-package".

Stáhneme si jádro z www.kernel.org (http://www.kernel.org):

```
# cd /usr/src
# wget ftp://ftp.funet.fi/pub/linux/kernel/v2.4/linux-2.4.20.tar.bz2
# tar -xjvf linux-2.4.20.tar.bz2
```

Potom stáhneme záplatu:

```
# wget ftp://ftp.solucorp.qc.ca/pub/vserver/patch-2.4.20ctx-17.gz
```

Zalepíme jádro se staženou záplatou

```
# cd linux-2.4.20
# zcat ../patch-2.4.20ctx-17.gz | patch -p1
```

Zjistíme/Ověříme si které jádro práve používáme:

```
# uname -r
# cat /proc/version
```

Zkopírujeme si aktuální konfiguraci ke zdrojům nového jádra:

```
# cp /boot/config-2.4.19 .config
```

Poopravíme konfiguraci podle nového jádra:

```
# make menuconfig
# make dep
```

A vytvoříme balíček s novým jádrem

```
# make-kpkg binary
```

Vytvořený balíček nainstalujeme například pomocí příkazu

```
# dpkg -i nový-balíček-s-jádrem.deb
```

8.2.13. Jednoduché úlohy

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

8.2.13.1. Přepnutí se do virtuálního serveru

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
# vserver název-vsirtuálního-serveru enter
```

8.2.13.2. Přemístnění serveru na jiného hosta

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
# rsync -vazPxle ssh --numeric-ids --delete \
       --exclude log --exclude nslog --bwlimit=100 --hard-links
       someserver.firma.net:/vservers/ /server/rs1
```

You can also replace all `*/*/*/*/log`. by a simple `*/log`

```
# rsync -vazPxle ssh --numeric-ids --delete \
 --exclude=*/log   \
 --exclude=*/proc \
 --exclude=*/dev   \
 --exclude=*/nslog \
 --bwlimit=100 --hard-links  ${SOURCE} ${DEST}
```

8.2.13.3. Instalace MTA

* *rcsinfor="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Jako MTA používám na většině virtuálních strojů program ssmtpd. Pro jednoduché účely, kdy vyžadujeme jen možnost odeslat poštu, úplně stačí.

Program tedy nainstalujeme

```
# apt-get install ssmtp
```

V rámci konfigurace zadámě „nějaké“ parametry a poté opravíme, nebo spíše přepíšeme konfigurační soubory. Do souboru `/etc/ssmtp/ssmtp.conf` jsem napsal:

```
***     This file is auto-generated using debconf on install.     ***
*** Any changes made may be overwritten on next upgrade depending ***
***  on your answer to the question on overwriting config files.  ***
#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
root=radek@hnilica.cz

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=localhost

# Where will the mail seem to come from?
rewriteDomain=hnilica.cz

# The full hostname
hostname=deb.hnilica.cz

# Set this to never rewrite the From: line (unless not given) and to
# use that address in the from line of the envelope.
FromLineOverride=YES
```

Kde `deb` v `hostname` je jméno virtuálního stroje.

Druhý konfigurační soubor `/etc/ssmtp/revaliases` jsem ponechal beze změn, tedy prázdný.

8.2.14. Nezapracované texty

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

8.2.14.1. Přesun virtuálního serveru na jiný host

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
# rsync -aPxvze ssh --delete --stats {,beta:}/vservers/uno/
```

8.2.14.2. Jednoduchý obraz disku pro malý virtuální server

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Stáhl jsem si soubor `TEST_32M_public.img`

```
# wget http://vserver.13thfloor.at/Stuff/QEMU/TEST_32M_public.img.bz2
```

```
no, this is a disk image, including a partition table,
with qemu you use it like this:

 qemu-fast -nographic -m 128 -snapshot -hda TEST_32M_public.img

so you have to use the offset of the first partition
for a loop setup (63*512 = 32256)

HTH,
Herbert
```

8.2.15. Problémy

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

8.2.15.1. hostname

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
On Tuesday 24 June 2003 21:17, Chris Murton wrote:
> vserver-1:/# hostname host-28
> hostname: you must be root to change the host name
> vserver-1:/# whoami
> root
>
> Any thoughts? ;)
>
Any original vserver have this bug.
I fix it in snapshot rh-vserver-1052304359.diff.gz (7 may).
```

```
For fix you must change one lines in kernel/sys.c.
From
asmlinkage long sys_sethostname(char *name, int len)
{
        int errno;
        char *nodename;

        if( (!capable(CAP_SYS_ADMIN)))
                return -EPERM;
To
asmlinkage long sys_sethostname(char *name, int len)
{
        int errno;
        char *nodename;

        if( (!capable(CAP_SYS_ADMIN)) && ! is_vps_admin())
                return -EPERM;

Also need fix sys_setdomainname.

--
With best regards,
Alex
```

8.2.15.2. sendmail

Odkazy a zdroje:

- Installing sendmail & Co. on Debian/Vserver (http://lena.franken.de/linux/debian_and_vserver/sendmail.html)
- . ()

8.3. User Mode Linux

Odkazy a zdroje:

- Virtuální linuxový server — User Mode Linux (http://www.abclinuxu.cz/clanky/show/72194)
- The User-mode Linux Kernel Home Page (http://user-mode-linux.sourceforge.net/)
- …

FIXME:

8.4. Free VPS

Free Virtual Private Server Solution (http://www.freevps.com/)

Let us to announce some new features in FreeVPS 1.3:
1. CPU Limit - allow to set up the upper limit of CPU usage inside VPS
2. CPU QoS - allow to set up the lower limit of CPU resource available inside VPS
3. Restore RSS memory accounting

4. DiskQuota speed optimization - separating dquota hash per each context
5. init emulation - add teinit, reboot, halt tools

8.5. QEMU

Odkazy:
- QEMU CPU Emulator (http://fabrice.bellard.free.fr/qemu/)
- QEMU (http://wiki.debian.org/QEMU) na Debian Wiki
-
-
-

8.5.1. Instalace na Debian Lenny

Odkazy:
- QEMU # Lenny (http://wiki.debian.org/QEMU#Lenny) na Debian Wiki
-

```
# aptitude install kvm
```

Zavedení mosulu kvm_intel bylo neúspěšné.

```
# /etc/init.d/kvm restart
Module kvm_intel not loaded failed!
FATAL: Error inserting kvm_intel (/lib/modules/2.6.26-2-amd64/kernel/arch/x86/kvm/kvm-intel.ko
Module kvm_intel failed to load failed!
```

Bylo třeba povolit v BIOSu virtualizaci.

```
# aptitude install qemu qemuctl qemu-launcher
```

8.6. WMvare

Emulátor počítače s procesorem Intel. Emuluje celý hardware takže je v něm možno spustit jakýkoliv operační systém. Jedná se o komerční produkt dostupný pod platformy: Linux, MS Windows, **FIXME:**doplnit ...

Kapitola 9. Uživatelé a skupiny

UNIX je multiuživatelský operační systém. Musí mít tedy způsob jak od sebe jednotlivé uživatele odlišit. Odlišit nejen je ale i jejich soubory. K tomuto odlišení se používá jedinečné číslo kterým je uživatel identifiková, uid (*user id*). Toto číslo je v rozsahu od 0 do maximálního počtu uživatelů. Tento maximální počet je dán systémem.

Uživatel s uid 0 je speciální uživatel, tak zvaný superuživatel, v UNIXu nazývaný root. TEnto superuživatel má nadstandardní práva a smí vykonávat administrátorské činnosti a jiné práce, které nejsou běžným uživatelům povoleny.

Aby si uživatelé nemuseli pamatovat svá uživatelská čísla, máme možnost používat přihlašovací jména. Tato jsou uložena v souboru `/etc/passwd` a jednoznačně přiřazují každému přihlašovacímu jménu uživatelské číslo.

9.1. Správa uživatelských účtů (kont)

* *chapter id="account-management" xreflabel="Správa uživatelských účtů"*

Ačkoliv můžeme velmi snadno zakládat a rušit uživatele „ručně“, tedy editací souboru `/etc/passwd` (a `/etc/shadow`) a ručním vytvářením či odstraňováním domovských adresářů, pro ulehšení existují programy které nám v tom pomohou. Jedná se o programy: **useradd**, **userdel** a **usermod**. Tyto programy po řadě provádí zakládání uživatelských účtů, rušení uživatelských účtů a změny uživatelských účtů.

Jedná se o základní programy, které jsou společné většině distribucí Linuxu a jejich obdoba by se měla nacházet na každém unixu.

V některých distribucích Linuxu, například v Debianu je k dispozici balíček `adduser` s programy **adduser** a **deluser**. Tyto programy jsou sofistikovaněší a pokud máme v systému více uživatelů, doporučuji je používat místo **useradd** a **userdel**.

Úpravou konfiguračních souborů `/etc/adduser.conf` a `/etc/deluser.conf` dosáhneme změny chování těchto programů.

Zde zmíním několik parametrů těchto konfiguračních souborů:

`DHOME`=/home

: Adresář který obsahuje uživatelské adresáře. Standardně `/home`. Pokud máme složitější adresářovou strukturu a nechceme zakládat uživatele přímo v adresáři `/home` můžem zde uvést `/home/users`. Nebo můžeme mít uživatelské adresáře v `/users`. Záleží na nás a naší představě.

`GROUPHOMES`=no

: Toto nastavení říká, jestli se adresáře pro uživatele mají vytvářet přímo v adresáři DHOME, nebo jestli se použije dvouúrovňová struktura DHOME/*groupname*/*username*. Standardní nastavení je `no` což znamená že uživatelské adresáře se vytváří přímo v DHOME jako `$DHOME`*username*. Tuto volbu využijeme pokud máme velké množství uživatelů a chceme jejich domovské adresáře strukturovat do skupin. Například pokud spravujeme školní server, můžeme vytvářet skupiny jako jednotlivé třídy žáků. To nám umožní snadno identifikovat a zrušit účty studentům, kteří dokončili studium.

`LETTERHOMES`=no

: Tato volba říká, jestli se mají uživatelské adresáře vytvářet v dvouúrovňové hierarchii, kdy pro všechny uživatelské adresáře uživatelů začínajících na stejné písmeno například „r“ jsou uloženy v adresáři `$DHOME/r`. Tato volba je podobná jako předchozí volba `GROUPHOMES` s tím rozdílem že klíč podle kterého se uživatelské adresáře združují je první písmeno názvu uživatele (uživatelského/přihlašovacího jména).

`FIRST_SYSTEM_UID`=100
`LAST_SYSTEM_UID`=999

Rozsah uživatelských čísel vyhrazených pro systémové uživatele. Tady se myslí systémoví uživatelé, zakládaní námi jako správci. Uživatelská čísla od 0 do 99 se používají pro účty jenž si zakládají například některé programové balíčky a číslo 0 jak jsem se již zmínil je superuživatel `root`.

`FIRST_UID`=1000
`LAST_UID`=29999

Rozsah uživatelských čísel vyhrazených pro běžné uživatele. Horní limit si můžeme upravit podle možností našeho systému. Nastavení těchto hodnot odráží naši politiku přidělování uživatelských čísel, pakliže nějakou máme. V jedné firmě jsem použil pro uživatele čísla tvaru 10000+*`osobní-číslo`* takovýto postup mi umožňoval udržet si pořádek v uživatelských souborech na všech serverech, protože na všech měl uživatel stejné uid.

`USERGROUPS`=yes
`USERS_GID`=100

Parametr `USERGROUPS` řídí přidělování uživatelů do skupin. Pokud je nastaven na `yes` je pro každého uživatele vytvořena skupina stejného jména a s identifikačním číslem skupiny gid stejným jako je číslo uživatele uid. Tento způsob vytváření uživatelských skupin nám umožňuje řídit přístupy k souborům s větší jemností. Pokud parametr nastavíme na `no` jsou všichni uživatelé automaticky přiřazení do skupiny `USERS_GID`, pokud jim není v příkazu **adduser** zadána jiná skupina.

`REMOVE_HOME`=0

Tento parametr (ze souboru `/etc/deluser`) říká, zdali se při zrušení uživatelského účtu má odstranit jeho domovský adresář se všemi soubory.

`REMOVE_ALL_FILES`=0

Parametr který určuje zdali se mají vyhledat všechny soubory rušeného uživatele a odstranit ze systému.

`BACKU`=0
`BACKU_TO`= "/backup/removed-users-files

V případě že je v předchozích volbách nastaveno odstraňování uživatelských souborů, tyto dva parametry určují jestli se má před odstranění souborů provést jejich záloha a když tak, kam se má provést.

* *Změnit název a strukturu kapitoly. Název: „Uživatelské učty“. Sestávat bude z částí: Co to je uživatelský učet (úvod), Co všechno k uživatelskému účtu patří (popis), Skupiny uživatelů, Správa účtu a skupin.*

V této kapitole se budeme věnovat správě uživatelských účtů. Popíšeme si co jsou to uživatelské účty, co vše k nim patří, jak se zkládají, ruší a upravují.

* ***FIXME:*** *Popsat co je to uživatelský účet a jeho náležitosti.*

Informace o účtech a skupinách jsou v souborech

`/etc/passwd`

FIXME:

Struktura jednoho řádku souboru

`jméno`:*`hash hesla`*:*`uid`*:*`gid`*:*`gecos`*:*`home directory`*:*`shell`*

Ukázka:

```
student:x:1000:100:Jan Novak:/home/student:/bin/bash
```

Obsahy jednotlivých polí jsou následující:

jméno

jméno uživatele, tzv. login jméno. Doporučuje se o délce nejvýše 8 znaků, sestávající pouze s písmen a číslic, začínající písmenem.

.

.

Pole shell obsahuje platný shell, nebo **/bin/true** či **/bin/false**. Seznam platných shellů se nachází v souboru `/etc/shells`:

- /bin/sh —
- /bin/true — nefungují terminálové služby mimo ftp
- /bin/false — žádné terminálové služby

`/etc/shadow`

FIXME:

`/etc/group`

FIXME:

skupina:hash hesla:číslo:seznam příslušníků skupiny odělených čárkou

`/etc/gshadow`

FIXME:

Základní nástroje pro práci s uživatelskými účty se nacházejí v balíčku `passwd`. Tyto nástroje stačí pro veškerou manipulaci s účty. Máme-li ovšem uživatelů více, oceníme balíček `adduser`. Tento přináší několik nových programů jenž usnadňují práci s účty poku máme těchto větší množství a potřebujeme sofistikovanější nástroje.

9.1.1. Založení účtu pro nového uživatele

Pro zakládání nových účtů mame k dispozici dva příkazy: **adduser** a **useradd**. **useradd** je vlastní program který provádí vytváření účtů.

FIXME:

* *Program **useradd** je součástí balíčku `passwd`. Program **adduser** pochází ze samostatného balíčku `adduser`. Tento balíček dále obsahuje i programy **deluser**, **addgroup** a **dellgroup**. Konfigurace je v souboru `/etc/deluser.conf`*

9.1.2. Zrušení účtu uživatele

Nejdříve zamezíme uživateli přihlásit se k počítači. Toto provedme odstraněním informací o uživateli ze souborů `/etc/passwd` (a `/etc/shadow`), `/etc/group` a (`/etc/gshadow`). Pro tento případ máme program userdel. Ukázkově si odstraníme uživatele *redviolet*

```
# userdel -r redviolet
```

Poté se podíváme do adresáře `/home`, zdali se tam nenachází ještě domovský adresář našeho uživatele. Neměl by, neb volba *-r* příkazu **userdel** tento odstraňuje. Ale protože já mám domovské adresáře jako symbolické odkazy, musím je rušit ručně. Poté si ještě prohlédneme celý souborový systém, zda-li se tam nenacházejí ještě nějaké soubory jenž náš uživatel vlastní (vlastnil).

```
# find / -uid 1029
```

Toto pak musíme vyřešit ručně, například smazáním souborů či předání jejich majiteli.

> **Varování**
>
> Číslo uživatele *uid* si musíme zjistit ještě před jeho odstraněním příkazem **userdell**.

9.1.3. Přidání uživatele do existující skupiny

Tato operace je velmi jednoduchá

```
# adduser uživatel skupina
```

9.1.4. Vytvoření uživatele ve skupině

```
# adduser --ingroup skupina uživatel
```

Kapitola 10. sudo

Pokud potřebujeme spouštět programy s jinými než vlastními právy. Například s právy roota, můžeme použít program sudo. Je to snažší a bezpečnější než se pokaždé přihlašovat jako root. Abychom ale mohli potřebný program spouštět, musíme si jej povolit v konfiguračním souboru `/etc/sudoers`. Úpravy tohoto souboru provádíme jako root příkazem **visudo**.

Kapitola 11. Sledování změn v systému

11.1. filetraq

Program umožňuje jednoduché sledování změn v konfiguračních souborech.

```
# aptitude install filetraq
```

Velmi užitečné jsou prográmky pro sledování změn v konfiguraci. Plní dvojí úlohu.

Pokud administrujeme stroj sami, dostáváme mailem všechny úpravy co jsme provedli. Tedy víme kdy jsme jakou úpravu provedli, a můžeme se postupně krok po kroku vrátit zpět.

Pokud je administrátorů víc, jsou všichni informováni o proběhlých změnách. Protože sledování změn v konfiguraci je samostatný program či skupina programů, nemůže se stát že zapomeneme ostatní administrátory informovat o tom co jsme upravovali.

Kapitola 12. Instalační balíčky a jejich systémy a repositáře

12.1. Debianí DEB balíčky

Odkazy:
- HOWTO: Install downloaded .DEB packages (and their dependencies) in 2 steps (http://ubuntuforums.org/showthread.php?t=62943)
- The Debian GNU/Linux FAQ Chapter 7 - Basics of the Debian package management system (http://www.debian.org/doc/FAQ/ch-pkg_basics.en.html)

Pokud potřebujeme nainstalova deb balíček můžeme tak učinit pomocí dpkg, což je nástroj nejnižší úrovně pro práci s deb balíčky.

```
# dpkg -i balíček.deb
```

Takto ovšem nejsou vyřešeny závislosti. Ty můžeme opravit následně příkazem

```
# apt-get -f install
```

Nebo příkazem

```
# aptitude -f install
```

Místo uvedeného dvoupříkazového postupu můžeme použít příkaz **gdebi**. Tento je součástí balíčku `gdebi`. Ovšem instlace tohoto balíčku si přítáhne asi 28MB závislostí na Pythonu a X11. Gdebi má grafický ekvivalent **gdebi-gtk**.

12.1.1. Personal Builder (pbuilder)

Zdroje a odkazy:
- pbuilder User's Manual (http://www.netfort.gr.jp/~dancer/software/pbuilder-doc/pbuilder-doc.html)
-

Program pro automatické sestavování balíčků.

Pbuilder je nahrazen programem `sbuild`.

12.1.2. reprepro

Zdroje a odkazy:
- reprepro (http://mirrorer.alioth.debian.org/) site
- Upravit Balíčky pro Debian - 11 (repozitáře a upload do distribuce) (http://www.abclinuxu.cz/clanky/navody/balicky-pro-debian-11-repozitare-a-upload-do-distribuce)
- Setting up and managing an APT repository with reprepro (http://www.jejik.com/articles/2006/09/setting_up_and_managing_an_apt_repository_with_reprepro/)
- HOWTO create and maintain a repository using reprepro and DeBaBaReTools (http://my.opera.com/atomo64/blog/howto-create-and-maintain-a-repository-using-reprepro-and-debabaretools)

- Setting up a basic Debian repository with reprepro (https://noc.sidux.com/fll/wiki/reprepro) — nutno schválit zabezpečení
- Repository pomocí reprepro (http://www.abclinuxu.cz/blog/nijel/2006/5/repository-pomoci-reprepro)

Nejdříve nainstalujeme balíček `reprepro`.

```
# aptitude install reprepro
```

Nyní učiníme rozhodnutí, kde budeme mít náš repositář. Doporučuji použít adresář `/var/packages`, ale na stroji kdy /var byl na melém disku a nebylo lze to řešit jinak jsem použil adresář `/home/packages`. Vytvoříme adresář a v něm podadresář s konfiguračními soubory. Tento se jmenuje `conf`.

```
# mkdir -p /var/packages/conf
```

Vytvoříme dva konfigurační soubory. Můžete se inspirovat mými.

Příklad 12-1. conf/distributions

```
Origin: Radek Hnilica
Label: Hnilica
Suite: stable
Codename: lenny
Architectures: i386 amd64 source
Componentes: main contrib non-free
Description: Balicky pro NB DELL D820, vanor
```

Příklad 12-2. conf/incoming

```
Name: incoming
IncomingDir: /var/packages/incoming
TempDir: /var/packages/tmp
Allow: stable>lenny
Default: lenny
```

Mimo dva programy **changestool** a **reprepro** a jejich manuálové stránky, se nainstaluje dokumentace do adresáře `/usr/share/doc/reprepro`

`distributions`

```
wimp:/var/packages/conf# cat distributions
```

`incoming`

Add debian packages to your repository

```
$ reprepro -Vb . includedeb etch-unstable /where/your/debs/live/*
```

Remove debian packages from your repository

```
$ reprepro -vb . remove etch-unstable weblion-package
```

12.1.2.1. Publikace vytvořené repository

Zdroje a odkazy:

- Setting up and managing an APT repository with reprepro (http://www.jejik.com/articles/2006/09/setting_up_and_managing_an_apt_repository_with_reprepro/)
-

-
-
-

```
# aptitude install apache2
```

12.1.2.2. Jednoduché úkony

12.1.2.2.1. Přegenerování/Vyvtoření indexových souborů

reprepro -b $BASEDIR export $CODENAMES

```
# reprepro -b /var/packages export lenny
```

12.1.2.2.2. Import balíčků

Balíčky nakopírujeme do adresáře `incoming` v našem repositáři a spustíme příkaz:

```
# reprepro -b /var/packages --waitforlock 100 processincoming incoming
```

12.2. Standardí repositáře Debianích balíčků

Odkazy:

- Running older (pre-etch) Debian releases (http://blogs.igalia.com/aperez/2009/11/running-older-pre-etch-debian-releases/)
-

Konfigurace repositářů se nachází v souboru `/etc/apt/sources.list`. Zde je každý repositář reprezentován jedním řádkem. Repositáře pro Lenny mohou vypadat třeba takto:

```
deb http://ftp.cz.debian.org/debian/ lenny main non-free contrib
deb-src http://ftp.cz.debian.org/debian/ lenny main non-free contrib
deb http://security.debian.org/ lenny/updates main non-free contrib
deb-src http://security.debian.org/ lenny/updates main non-free contrib
deb http://volatile.debian.org/debian-volatile lenny/volatile main
deb-src http://volatile.debian.org/debian-volatile lenny/volatile main
deb http://www.backports.org/debian lenny-backports main contrib non-free
deb http://www.debian-multimedia.org lenny main
```

Pokud máme někde starý stroj, s distribucí která už dávno vymizela ze standardních repositářů, s výhodou využijeme archivu repositářů.

```
deb http://archive.debian.org/debian sarge main contrib non-free
deb http://archive.debian.org/debian-security sarge/updates main contrib non-free
deb http://archive.debian.org/backports.org sarge-backports main contrib non-free
```

Kapitola 13. Server configuration tools

Zdroje a odkazy:

- cfengine2 ()
- Bcfg2 (http://trac.mcs.anl.gov/projects/bcfg2/)
-
-
-

Programy a balíčky

- clusterssh — administer multiple ssh or rsh shells simultaneously
- csync2 — cluster synchronization tool
- dish — the diligence/distributed shell for parallel sysadmin
- facter — a library for retrieving facts from operating systems !!!
- genders — cluster configuration management database tools
- timeout — un a command with a time limit. (Lenny: 1.11-6.5)
-
-
-

Kapitola 14. Server monitoring

Zdroje a odkazy:

- Linux server monitoring (http://ehsanakhgari.org/blog/2008-05-20/linux-server-monitoring)
- PIKT Global-View, Site-at-a-Time System and Network Administration (http://pikt.org/pikt/links.html)
-
-

Programy

- mon
- monin
- MONIT (http://mmonit.com/monit/) a M/MONIT (http://mmonit.com/)
- Spong (http://spong.sourceforge.net/) — System and Network Monitoring (poslední verze 2.8.0-beta_2 z 2005-09-30)
- event monitor (http://www.jmcresearch.com/projects/eventmonitor/) (19/Nov/1999 version 0.6r4)
- BigSister (http://www.bigsister.ch/bigsister.html)
- The Arusha Project home page (http://ark.sourceforge.net/) (April, 2005)
- pica — System administration program similar to PIKT (0.4.1-9 Lenny)
- puppet — centralised configuration management for networks (0.24.5-3 Lenny)
- slack configuration management program for lazy admins (0.15.2-3 Lenny)
-
-

14.1. mon

* *section id="mon"*

Program mon slouží ke sledování dostupnosti sítě a jejích služeb. Základním výstupem jsou volání alarm skriptů. Je implementováno několik skriptů posílajících alarmy emailem, pagerem, zapisující do deníku, ... Další skripty je možno dopsat dle potřeby. Stejným způsobem, tedy skripty jsou řešeny i kontroly jednotlivých typů služeb. V základní sadě je ping, http, ftp, dns, …

Instalace na Debian Woody je velmi jednoduchá

```
# apt-get install mon
```

Balíček `mon`(0.99.2-2) si v rámci závislostí přiinstaloval několik perlovských knihoven (`libconvert-ber-perl`(1.31-1) `libmon-perl`(0.11-2) `libtime-hires-perl`(1.20-4) `libtime-period-perl`(1.20-7))

K základnímu balíčku je třeba doinstalovat řadu dalších, podle použitých monitorů služeb. U popisu monitorů tyto balíčky uvedu.

Konfigurační soubory se nacházejí v adresáři `/etc/mon`. Jsou to tyto soubory:

`auth.cf`

FIXME:

`mon.cf`

FIXME:

Dále se zde ještě nachází adresář `monshow`.

K základnímu balíčku mon jsem přiinstaloval balíčky:

- `ping`
- `libnet-dns-perl`

Příklad 14-1. Ukázka částí konfiguračního souboru `/etc/mon/mon.cf`

```
alertdir        = /usr/lib/mon/alert.d      [co id="co.alertdir"/]
mondir          = /usr/lib/mon/mon.d        [co id="co.mondir"/]
maxprocs        = 10
histlength      = 100
randstart       = 15s

watch brno.example.cz
    service ping
        description Border router
        interval 5m
        monitor fping.monitor
        period wd {1-7}
            alertevery 30m
            alert netpage.alert carlito@example.com
            alert mail.alert jose@example.com

...

watch fors.example.cz
    service ping
    ...
    service www
        description WWW server
        interval 5m
        monitor http.monitor
        depend fors.example.cz:ping
        period wd {1-7}
            alertevery 30m
            alert netpage.alert carlito@example.com
            alert mail.alert jose@example.com

...

watch users
    service ping
        service End point routers
        interval 5m
        monitor fping.monitor
        depend hodonin.example.cz:ping
        period wd {1-7}
            alertevery 30m
            alert netpage.alert carlito@example.com
            alert mail.alert jose@example.com
```

14.1.1. Programy

V balíčku `mon` je mimo vlastní démon `mon` i řada dalších programů. Jsou to:

moncmd

FIXME:Řádkový klient pro komunikaci s mon démonem `mon`.

monshow

FIXME:Řádkové a CGI rozhraní

skymon

FIXME: Dvousměrné rozhraní pro SkyTel pagery. Umožňuje komunikovat d `mon` démonem podobně jako program **moncmd** s použitím vašeho obousměrného pageru. Mělo by jít vyoužít přes SMS.

mon.cgi

FIXME: Interaktivní webové rozhraní umožnující nejen zjišťovat stav služeb ale i měnit nastavení a parametry `mon` démona.

14.1.1.1. moncmd

FIXME:

S bežícím démonem `mon` je možno komunikovat pomocí programu **moncmd**.

14.1.2. Monitory služeb

- `fping` — používá jej `fping.monitor`

14.1.2.1. fping.monitor

Monitor `fping.monitor` potřebuje ke své činnosti program **fping**. Ten se nachází v balíčku stejného jména.

Příklad 14-2. Příklad užití monitoru `fping.monitor`

```
watch joshua.example.cz
    service fping
        description fping joshua
        interval 5m
        monitor fping.monitor
        period wd {1-7}
            alertevery 15m
            alert mail.alert radek@example.cz
            upalert mail.alert -u radek@example.cz
```

14.1.2.2. ldap.monitor

FIXME: Monitor `ldap.monitor` potřebuje ke své činnosti perlovskou knihovnu `LDAP.pm` jenž je v balíčku `libnet-ldap-perl`.

14.2. Monitorování serveru pomocí munin a monit

Odkazy a zdroje:

- Server Monitoring With munin And monit (http://www.howtoforge.com/server_monitoring_monit_munin)
- Monitor Your Linux Server With Munin (http://tombuntu.com/index.php/2007/07/16/monitor-your-linux-server-with-munin/)
- MUNIN (http://munin.projects.linpro.no/)
-

14.3. Monit

- Ubuntu / Debian Linux: Install Monit Linux Server Monitoring Utility (http://www.cyberciti.biz/faq/monit-linux-server-monitor-software-installation/)

Poznámka: M/Monit je komerční software ze peníze.

Kapitola 15. Zajímavé programy pro administrátory

V této kapitole jsou zmínky o řadě programů jenž si z hlediska administrátora zaslouží pozornost. Mnohé z nich vám ušetří prác s programováním vlastního nástroje.

- launchtool (http://people.debian.org/~enrico/launchtool.html) — Runs a command supervising its execution
- mrb — Manage incremental data snapshots with make/rsync
-
-

Kapitola 16. Ukládání a organizace dat

Šablona pro nové kapitoly

Informace na UNIXu jsou ukládány do diskového prostoru. Jsou ukládány v jednotkách zvaných soubory. Soubor je souvislá posloupnost znaků (bajtů). Soubor je popsán metainformacemi které slouží k orientaci mezi soubory. Jednou z metainformací je název souboru, přes který soubor vyhledáváme, identifikujeme, přistupujeme k němu. Soubory jsou organizovány do hierarchické adresářové struktury zvané strom. V této struktuře jsou dána pravidla jimiž se umístnění souboru řídí.

16.1. Adresáře

Adresáře jsou způsob organizace souborů. Každý soubor můsí být/přináležet do nějakého adresáře. Adresářová struktura je hieararchická, adresář sám patří do nějakého jiného adresáře.

Kořenový adresář, označovaný `/` je jediný adresář v systému jenž nepatří do žádného jiného adresáře a je ve stromové hierarchii předkem všech souborů a adresářů.

V pojmenování souborů a adresářů je řád jenž se pokusím dále popsat. Začnu adresáři.

Jména adresářů

`bin`

: Jméno bin se používá pro adresáře jenž opsahují spustitelné/vykonatelné programy (binárky) či skripty. V adresářích s tímto jménem by se neměly vyskytovat soubory jiného typu.

`etc`

: V tomto adresáři jsou uložené konfigurace programů a systému.

`lib`

: Toto jméno se používá v souvislosti s knihovnami (*library*). Adresář pak obsahuje knihovní soubory statických či dynamických knihoven. Obsahuje taky knihovny funkcí interpretovaných jazyků jako jsou Perl, Tcl/TK, . . .

`sbin`

: Význam adresáře `sbin` je obdobný jako u adresáře `bin`. Rozdíl spočívá ve významu programů v něm uložených. V adresáři `sbin` jsou programy jenž jsou důležité pro správu/údržbu systému a běžní uživatelé jej nepoužívají či nemají právo používat.

`src`

: V adresáři tohoto jména bývají zdrojové kódy programů systému, jádra i uživatelských programů. Jeho název je odvozen od slova *source* (zdroj).

`tmp`

: Tento adresář slouží pro ukládání dočasných (*temporary*) pracovních souborů. Často bývají tyto adresáře pravidelně čišteny/promazávány.

`var`

: Název `var` se používá pro adresáře v nichž se data/soubory často mění. Bývají to různé databáze, deníky, úložiště dat.

.

FIXME:

Uvedená seznam jmen adresářů není vyčerpávající, ale podchycuje ta nejdůležitější. Nyní již k samotné stromové hierarchii adresářů. Kořenový adresář `/` obsahuje následující adresáře:

`/bin`

Obsahuje důležité programy pro práci se systémem jenž jsou nezbytné při administraci systému. Jedná se o pprogramy které mohou používat všichny uživatelé a jejichž přítomnost v systému je nezbytná pro základní údržbu.

`/boot`

V tomto adresáři bývají soustředěny všechny soubory nutné pro zavedení systému při startu počítače. Jedná se zejména o obraz jádra operačního systému. Obraz init ramdisku. Svá data zde také ukládá zavaděč grub do vlastního adresáře `/boot/grub`.

`/dev`

V tomto adresáři jsou soubory zařízení. Jedná se o speciální soubory jenž reprezentují jednotlivé části hardware a slouží pro kominikaci s těmito. Jsou zde například speciální soubory `/dev/hd...` reprezentující IDE disky, `/dev/tty...` reprezentující sériové linky a virtuální terminály a mnoho dalších.

`/etc`

V tomto adresáři jsou sostředěny konfigurace veškerého programového vybavení v systému a také systému samotného.

`/home`

Zde bývají zakládány domovské adresáře uživatelů systému.

`/lib`

Knihovny důležité pro samotný běch systému. Taktéž jsou zde adresáře `/lib/modules` jenž obsahuje moduly zaveditelné do jádra a `/lib/security` s moduly systému PAM.

`/opt`

Adresář `/opt` bývá kořenovým adresářem pro komerční programové vybavení.

`/proc`

Tento **virtuální** adresář je rozhraním jádra přes které jádro publikuje řadu informací a je možno ovlivňovat jeho fukčnost.

`/sbin`

Adresář obsahuje důležité programy pro administraci systému jenž nejsou potřebné pro běžné uživatele.

`/sys`

Virtuální adresář `/sys` rozhraním do jádra operačního systému. Tento adresář má nahradit adresář `/proc`.

`/tmp`

Prostor pro vytváření dočasnch souborů. Tento adresář je nutný pro běh systému, a svazek na kterém se nachází musí mít sdostatek volného prostoru.

`/usr`

Tento adresář je kořenem hierarchie programových adresářů. Zde jsou nainstalovány programy, jejich knihovny i data. Adresář není nezbytný pro chod systému a může být připojován například s použitím 34 i z jiného počítače či ze vzdáleného serveru.

`/var`

Zde jsou soustředěny všechny aplikační data jenž se často mění. Příkladem takových dat jsou databáze, úložiště pro elektronickou poštu, zámky souborů, tiskové fronty a další.

16.2. Struktura adresářů

V kořenovém adresáři `/` jsou tyto podadresáře

Tabulka 16-1. Podadresáře kořenového adresáře

`bin`	Esential command binaries
`boot`	Static files of the boot loader
`dev`	Device files
`etc`	Host-specific system configuration
`home`	User home directories
`lib`	Essential shared libraries and kernel modules
`mnt`	Mount point for mounting a file system temporarily
`proc`	Virtual directory for system information
`root`	Home directory for the root user
`sbin`	Essential system binaries
`tmp`	Temporary files
`usr`	Secondary hierarchy
`var`	Variable data
`opt`	Add-on application software packages

The following is a list of important considerations regarding directories and partitions:

- The root partition `/` must always physically contain `/etc`, `/bin`, `/sbin`, `/lib` and `/dev`, otherwise you won't be able to boot. Typically 100MB is needed for the root partition, but this may vary.

Kapitola 17. PAM

Pluggable Authentication Modules

PAM je mechanizmus jenž zajišťuje, jak již z názvu vyplývá, autentikaci. Tedy ověření uživatele. Jedná se o knihovnu jenž programy potřebující ověřovat uživatele volají a tato pak přes konfiguraci realituje vlastní oveření pam moduly. Tato koncepce dovoluje provést sofistikovanou konfiguraci a rovněž autentifikovat metodami (moduly) které v době psaní programu ani nebyly známy.

17.1. Autentikace přes MySQL

Tato metoda používá modul `libpam-mysql` který provádí ověřování uživatele dotazy do MySQL databáze.

Kapitola 18. Správa energie

Šablona pro nové kapitoly

Odkazy a zdroje:

- ACPI4Linux at a glance (http://acpi.sourceforge.net/index.html)
- Centrino & Linux (http://www-lehre.inf.uos.de/~rfreund/acpi)
- Linux Power Management: APM/ACPI etc (http://rzr.online.fr/docs/comp/powerm.htm)
- ACPI: Advanced Configuration and Power Interface (http://www.tldp.org/HOWTO/ACPI-HOWTO/)
- Power Management with Linux - APM, ACPI, PMU (http://tuxmobil.org/apm_linux.html)
- Documentation: The /proc/acpi/processor Subdirectory (http://acpi.sourceforge.net/documentation/processor.html)

Slovníček pojmů a skratek

ACPI

Advanced Configuration and Power Interface

APM

Advanced Power Management

OSPM

Operating System’s Power Management

Stavy systému

C0

FIXME: running

C1

FIXME:

C2

FIXME: sleep

C3

FIXME: sleep

ACPI didn’t make it possible to break the OS’s reliance on the firmware, but it did make it possible to rely on it to a much lesser degree. The key way it did this was through the introduction of ACPI source language (ASL) and ACPI machine language (AML). ASL and AML allowed the firmware to communicate to the OS the steps necessary to perform actions on its platform, but then the OS was responsible for actually executing them.

18.1. Sekce

Non ACPI just gives you two speeds to work with, full 1.8 and slow 1.2.

```
# echo "1200000:1800000:powersave" > /proc/cpufreq
# echo "1200000:1800000:performance" > /proc/cpufreq
```

You can actually tweak the cpu through the acpi interface too:

```
# echo "N" > /proc/acpi/processor/CPU/throttling #where 0 <= N <= 7.
```

Kapitola 19. Dálková správa stroje

Přihlášení, kopírování souborů.

Protože s UNIXovými systémy pracujeme pře sériovou konzolu, tedy přes obyčejný sériový terminál. I konzole na Linuxu simuluje obyčejný sériový terminál. Dálková správa je tedy o tom, jak vytvořit spojení mez místem kde se nacházíme a mezi počítačem který chceme spravovat. V drívějších dobách se používaly telefonní linky a modemy. V dnešní době internetu používáme sitová spojení. Na počítačích běží služba která nám umožní se připojit klientským programem a získat tak terminálové připojení k počítači.

Existuje několik programů k tomu určených:

- rlogin
- telnet
- ssh

Některé z nich si blíže popíšeme.

19.1. ssh

* ***Attributy:*** *id="ssh"*

Odkazy a zdroje:

- Oficiální stránky standardu Bluetooth (http://www.bluetooth.org)

ToDo

1. Přečíst knihu „SSH kompletní průvodce“.
2. Popsat používaní tunelů skrze ssh
3. Rozepsat se o konfiguraci ssh démona

přihlášení

```
$ ssh uživatel@host
```

poté jsme vyzváni k zadání hesla. Po ověření tak získáme běžící shell na vzdáleném počítači.

Chceme-li vykonat jen jeden příkaz, můžeme použít

```
$ ssh uživatel@host příkaz
```

Pro potřeby „skriptování“ je potřeba vytvořit sobě klíč (klíčový pár). Soukromá část zustane na našem počítači s veřejnou uložíme na cílový počítač.

1. Vytvoříme si klíčový pár

   ```
   $ ssh-keygen
   ```

   Veřejná část se uloží do souboru `~/.ssh/identity.pub` a soukromá do `~/.ssh/identity`
2. Veřejnou část uložíme na cílový stroj

   ```
   $ scp .ssh/identity.pub host:.ssh/authorized_keys
   ```

   Lépe však použít příkaz **ssh-copy-id** který nás ochrání před chybami typu přepsání souboru s klíči a tím odstranění původně tam daných klíčů.

   ```
   $ ssh-copy-id username@host
   ```

Tabulka 19-1. Důležité soubory

název	obsah
`/etc/ssh/ssh_known_hosts`	
`$HOME/.ssh/known_hosts`	
`$HOME/.ssh/authorized_keys`	veřejné části asymetrických klíčů uživatelů kterří mají právo se hlásit ssh klíči

19.1.1. Přihlašování bez zadání hesla

19.1.1.1. Na Potato, SSH1

Na serveru nastavíme v `/etc/ssh/sshd_config`

```
RSAAuthentication yes
```

a na straně klienta si vytvoříme klíč

```
$ ssh-keygen
```

tento klíč přidáme do souboru na straně serveru

```
$ cat .ssh/identity.pub | ssh uživatel@host "cat - >>.ssh/authorized_keys"
```

19.1.1.2. Na Woody, SSH2

Na straně klienta si vytvoříme klíč

```
$ ssh-keygen -t rsa
```

tento klíč přidáme do souboru na straně serveru

```
$ cat .ssh/id_rsa.pub | ssh uživatel@host "cat - >>.ssh/authorized_keys"
```

Na straně klienta si ještě popravíme soubor `/etc/ssh/ssh_config`

```
HostbasedAuthentication yes
PreferredAuthentication hostbased,...
```

19.1.1.3. Nastavení serveru

Pokud existuje na serveru soubor `/etc/nologin`, sshd povolí přihlášení pouze uživateli root. Žádný jiný uživatel se ke svému účtu nedostane. Tedy pokud chceme rychle zamezit přihlašování uživatelů, stačí vydat příkaz

```
# touch /etc/nologin
```

Při autentizaci veřejným klíčem je potřeba nastavit v konfiguraci serveru

```
RSAAuthentication yes
```

případně

```
DSAAuthentication yes
```

Varování

Na straně serveru musí být u účtu správná nastavení přistupových práv na adresář `.ssh` a soubory v něm, jinak se nelze přihlásit s použitím klíčů.

```
$ ls -ld .ssh
drwx------    2 radek    radek        4096 čec 10 22:29 .ssh/
$ ls -l .ssh
celkem 4
-rw-r--r--    1 radek    radek         732 čec 10 22:22 authorized_keys
-rw-r--r--    1 radek    radek           0 čec 10 22:21 known_hosts
```

19.1.2. Použití ssh agenta `ssh-agent` ssh-agent

FIXME: dopsat

19.1.3. Authprogs - omezení SSH na vybrané příkazy

Odkazy a zdroje:

- The Authprogs SSH Command Authenticator (http://www.hackinglinuxexposed.com/articles/20030115.html)
- Authprogs - snadné spouštění akcí prostřednictvím SSH (http://www.linuxzone.cz/index.phtml?ids=29&idc=838)
- Program Authprogs (http://www.hackinglinuxexposed.com/tools/authprogs/src/)

Authprogs je skrip jenž nám umožňuje omezi uživatele na straně SSH serveru na spouštění jen vybraných programů s konkrétními parametry.

Příklad syntaxe souboru `~/.ssh/authprogs.conf`

```
[ ALL ]
        command0 arg arg arg

[ ip.ad.dr.01 ip.ad.dr.02 ]
        command1 arg arg arg

[ ip.ad.dr.03 ]
        command2 arg arg arg
        command3 arg arg
```

Příklad sprovoznění authprogs

1. Vygenerujeme klíče

   ```
   $ ssh-keygen -t dsa -f id_dsa
   ```

2. Do souboru `~/.ssh/authorized_keys` příslušného účtu na vzdáleném systému doplníme položku s klíčem

   ```
   command="/sbin/authprogs",no-port-forwarding,no-X11-forwarding,\
   no-agent-forwarding,no-pty ssh-dss AAAAB3Nzc......klíč...... == ja@mujstroj.firma.cz
   ```

3. Na vzdáleném systému nainstalujeme skript authprogs do adresáře `/sbin` a vytvoříme příslušný konfigurační soubor `~/.ssh/authprogs.conf`. Například takto:

```
[ 192.168.0.1 ]
uptime; df -k /home
who
```

19.1.4. Konfigurační soubory SSH

19.1.4.1. Autorizované klíče `~/.ssh/authorized_keys`

Soubor autorizovaných klíčů slouží k přihlašování se na účet bez nutnosti zadávat či dokonce znát heslo. Porovnává se veřejná část klíč v tomto souboru se soukromou částí kterou se prokazuje přihalšující se.

Struktura souboru je řádková. Co řádek to záznam. Na řádku jsou pak tyto informace:

`[parametry]` *`typ-klíče klíč identifikace_klíče`*

Příklad různých záznamů

```
ssh-rsa AAAAB3...vypuštěno...N8= radek@yoda
2048 35 271... vypuštšno...989 radek@yoda
command="/sbin/authprogs",no-port-forwarding ssh-dss AA...vypuštěno...= ja@mujstroj
```

Each RSA public key consists of the following fields, separated by spaces: options, bits, exponent, modulus, comment. Each protocol version 2 public key consists of: options, key type, base64 encoded key, comment. The options fields are optional; its presence is determined by whether the line starts with a number or not (the option field never starts with a number).

Zde uvádím seznam parametrů jenž můžeme použít. Parametry se oddělují čárkou a nesmí mezi nimi být mezery.

* *$ man 8 sshd*

Parametry a klíčová slova

from=*`specifikace_adresy`*

Slouží k omezení odkud se smí vlastník daného klíče přihlašovat. Příklad:

```
rom="*.umbc.edu"
```

`command="`*`příkaz`*`"`

Příkaz se vykoná místo příkazu uvedeného hlásícím se uživatelem. Tento parametr může být použit pro omezení uživatele jen na jeden příkaz. Využijeme například při zálohování.

`environment="`*`NAME=value`*`"`

FIXME:

`no-port-forwarding`

FIXME:

`no-X11-forwarding`

FIXME:

`no-agent-forwarding`

FIXME:

`no-pty`

FIXME:

`permitopen="host:port"`

FIXME:

19.1.5. Nezpracované texty

19.1.5.1. Forwardování portů pomocí SSH

* *Kopie clanku, NEPUBLIKOVAT!!!*

Odkazy a zdroje

- SSH a forwardování portů (http://poweroff.cz/index.php?clanek=19)

SSH může spojení pro jakýkoliv TCP/IP port přesměrovat na zabezpečený kanál pomocí port forwardingu. U port forwardingu můžeme buď namapovat lokální port na klientovi na vzdálený port na serveru či jakýkoliv port na serveru na jakýkoliv port na klientovi - čísla portů nemusí být shodná.

Pro vytvoření SSH kanálu který naslouchá na lokálním portu:

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

Tedy například pro stažení pošty ze serveru `mail.domain.com`:

```
ssh -L 1100:mail.domain.com:110 mail.domain.com
```

Poštu nyní můžete stahovat z localhostu připojením se na port 1100, poté bude vše forwardováno zabezpečeným kanálem na server mail.domain.com. Pokud nám na serveru mail.domain.com neběží SSH server, ale v síti, v které je mail.domain.com máme jiný server s SSH daemonem, můžeme forwardovat přes něj, tedy:

```
ssh -L 1100:mail.domain.com:110 other.domain.com
```

Stažení pošty ze serveru mail.domain.com z portu 1100 localhostu:

```
ssh -L 1100:mail.domain.com:110 mail.domain.com
```

Stažení pošty ze serveru mail.domain.com z portu 1100 localhostu přes server other.domain.com:

```
ssh -L 1100:mail.domain.com:110 other.domain.com
```

Zasílání pošty z portu 2525 localhostu na smtp mailserver.com s použitím komprimovaného SSH kanálu:

```
ssh -C -L 2525:mailserver.com:25 mailserver.com
```

Forwardování VNC spojení ze serveru vncserver.com na localhost s použitím komprimovaného SSH kanálu a s povolením vzdálených strojů připojování se na lokální forwardovaný port:

```
ssh -C -L 5901:vncserver.com:5901 vncserver.com -g
```

19.1.5.2. Přihlašování pomocí RSA/DSA klíče

```
PermitUserEnvironment Yes
```

Kapitola 20. Startování Linuxu

Popis procesu zavádění a spouštění OS Linux

Startovací seqvence jako procedura

1. krok

Startovací seqvence jako seznam

- zavadeci-seqvence-hardware

Zaváděcí seqvence hardware

- .

20.1. Startovací seqvence

.

20.2. Start jádra

1. Inicializace zařízení
2. Volitelné nahrání *initrd*
3. připojení kořenového souborového systému zadané v lilo, loadlin, nebo jiném zavaděči. Jádro vytiskne
   ```
   VFS: Mounted root (ext2 filesystem) readonly.
   ```
4. Je spuštěn program `/sbin/init` a získá číslo procesu (PID) 1. init vytiskne
   ```
   INIT: version 2.76 booting
   ```
   Přes parametr `boot=` zavaděče je možno spustit místo init jiný program.

20.3. Proces `/sbin/init`

`/sbin/init`

1. je přečten `/etc/inittab`
2. je spuštěn skript definovaný na řádce
   ```
   si::sysinit:/etc/init.d/rcS
   ```
3. init se přepne do *runlevel*u jenž je definován řádkou
   ```
   id:3:initdefault:
   ```

20.4. sysinit

debian: `/etc/init.d/rcS`

1. Spustí startovací skrypty `/etc/rcS.d/S*`
2. spustí `/etc/rc.boot/*` Nedoporučuje se!

redhat: `/etc/rc.d/rc.sysinit`

1. krok

Kapitola 21. Debian From Scratch (DFS)

* *chapter id="dfs" xreflabel="Debian From Scratch" condition="author"*

Odkazy a zdroje:

- Installing Debian From Scratch (http://www.newsforge.com/article.pl?sid=05/01/10/1727246)
- Odkaz ()

21.1. Nezpracované poznámky

FIXME: napsat text sekce.

Instalace základních balíčků:

```
# cdebootstrap sid /mnt file:///opt/packages
```

Nainstaluje balíčky z `/opt/packages` do systému jehož root je `/mnt`. Pomocí přepínače `-a` můžeme specifikovat architekturu.

```
# cdebootstrap -a amd64 sid /mnt file:///opt/packages
```

Program/příkaz **cdebootstrap** instaluje základní systém a nepoužívá k tomu nástroje jako **dpkg** nebo **apt**. Používá se jen v průběhu instalace, takže se s ním běžně nepotkáte.

Základní konfigurace síťového rozhraní se provede editací souboru `/etc/network/interfaces` do kterého vložíme

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dpcp
```

Kapitola 22. SGI

* *chapter id="sgi" condition="author"*

Odkazy a zdroje:

- Booting Mac OS X (http://www.kernelthread.com/mac/osx/arch_boot.html): Open Firmware, Bootloader, System Startup, BootCache
- Rozložení kláves Czech a Slovak by Loki 2.1.2 pro Mac OS X (http://lokisw.com/index.php?cat0=1#CzechSlovakByLoki)
- Technical Note TN2056, Installable Keyboard Layouts (http://developer.apple.com/technotes/tn2002/tn2056.html)
- **FIXME:**odkaz ()
- **FIXME:**odkaz ()

22.1. Instalace Indy R5000

.

```
Command Monitor.  Type "exit" or click on "done" to return to the menu.
>> boot stand/fx --x

# mkfs /dev/dsk/dks0d2s0

Command Monitor.  Type "exit" or click on "done" to return to the menu.
>> boot -f dksc(0,6,8)sashARCS dksc(0,6,7)stand/fx.ARCS --x
```

Kapitola 23. Počítače Macintosh fy Apple

* *chapter id="apple-macintosh"*

Odkazy a zdroje:

- Booting Mac OS X (http://www.kernelthread.com/mac/osx/arch_boot.html): Open Firmware, Bootloader, System Startup, BootCache
- Rozložení kláves Czech a Slovak by Loki 2.1.2 pro Mac OS X (http://lokisw.com/index.php?cat0=1#CzechSlovakByLoki)
- Technical Note TN2056, Installable Keyboard Layouts (http://developer.apple.com/technotes/tn2002/tn2056.html)

23.1. Poznámky a návody

FIXME:napsat text sekce.

23.1.1. Aktualizace stránky v prohlížeči Safari

Následující návod jsem napsal podle příspěvku Adama Nohejla z mailové konference <`konference@kppm.cz`>.

Pokud potřebujeme na straně prohlížeče aktualizovat stránku jejíž kód tuto automatickou aktualizaci (reload) neprovádí, můžeme náš prohlížeč Safari přimět ať periodicky aktualizuje stránku sám. Použití tohoto kódu je v případě kdy potřebujeme sledovat změny na cizím webu.

Otevřete si tedy editor skriptů `Applications/AppleScript/Script Editor` a napiště v něm následující kód

```
tell application "Safari"
    set doc to document of window 1
    repeat
        do JavaScript "window.location.reload(true)" in doc
        delay 15  -- UPRAVTE NA POŽADOVANÝ POČET SEKUND
    end repeat
end tell
```

Kód můžete uložit jako *stand-alone* applet.

23.2. Klávesnice a klávesnicové mapy

Odkazy a zdroje:

- KeyLayoutMaker: Perl script to create Mac OS X keyboard layouts (http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=keylayoutmaker)
- Technical Note TN2056 Installable Keyboard Layouts (http://developer.apple.com/technotes/tn2002/tn2056.html)
- Keyboard Builder (http://homepage.mac.com/poorant79/software/kb.html)
- M10L MAC Unleash Your Multilingual Mac (http://homepage.mac.com/thgewecke/pmlingos9.html)

Pro úpravu klávesnicových map můžeme použít program Ukelele (http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=ukelelehttp://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&ite

III. Sítě

Kapitola 24. TCP/IP

Síť na nejnižší úrovni

* *chapter id="tcpip"*

Pakete pakete, kampak si to míříš?

Zdroje a odkazy:

- A TCP Tutorial (http://www.ssfnet.org/Exchange/tcp/tcpTutorialNotes.html)

ToDo list

1. NetMap (http://netmap.sourceforge.net/) — Mapování sítě. Prozkoumat a zařadit.
2. Linuxové DMZ (http://www.abclinuxu.cz/clanky/ViewRelation?relationId=32207) — série článků na www.abclinuxu.cz (http://www.abclinuxu.cz)

Síťování v Linuxu se postupně vyvíjelo. Od nejstaršího až po dněšní.

Tabulka 24-1. Sťování v Linuxu

jádro	program
1.1.x	ipfw
2.0.x	ipfwadm
2.2.x	ipchains
2.4.x	iptables
2.6.x	nf

24.1. IP adresy

* *section*

Veřejné IP adresy

- `192.168.*.*`
- `172.16-31.*.*`
- `10.*.*.*`

24.1.1. Zvláštní/speciální IPv4 adresy

* *Výpisky z RFC3330 Special-Use IPv4 Addresses (http://www.rfc-editor.org/rfc/rfc3330.txt)*

`0.0.0.0/8`

Adresní blok odkazující na zdrojový stroj na „této“ síti. Adresa 0.0.0.0/32 může být použita jako zdrojová adresa pro tento počítač. Ostatní adresy mohou odkazovat na specifické počítače v lokální síti.

```
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
```

Soukromé adresy určené k použití v soukromých sítích. Tyto adresy se nesmí použít na internetu. Použití je definováno RFC1918 (http://www.rfc-editor.org/rfc/rfc1918.txt).

`14.0.0.0/8`

Tento blok je vyhrazen pro použití v mezinárodním systému veřejných datových sítí RFC1700 (http://www.rfc-editor.org/rfc/rfc1770.txt) strana 182. Čísla z tohoto bloku přiděluje IANA (http://www.iana.org/numbers.html).

`127.0.0.0/8`

Tento blok je vyhrazen pro použití jako loopback adresy. Datagram odeslaný na adresu v tomto bloku se vrací zpět na počítač. JAko konkrétní loopback adresa se používá 127.0.0.1/32. Žádná adresa z tohoto bleku se nesmí objevit na internetu. RFC1700 (http://www.rfc-editor.org/rfc/rfc1770.txt) strana 4.

`169.254.0.0/16`

„link local“ blok. Je vyhrazen pro komunikaci mezi stroji na jedné lince. Stroje dostávají adresy autokonfigurací, například když nemůže být nalezen DHCP server.

`192.0.2.0/24`

Tento blok je přiřazen jako „TEST-NET“ pro použití v dokumentaci a příkladech. Je často používán ve spojení s doménovým jménem `example.com`. Adresy z tohoto bloku se nesmějí objevit na internetu.

`224.0.0.0/4`

Blok adres dříve známých jak adresy třídy D. Je vyhrazen pro multicast adresy. Použití se řídí dokumentem RFC3171 (http://www.rfc-editor.org/rfc/rfc3171.txt)

`240.0.0.0/4`

Blok adres dříve známých jako adresy třídy E. Cílová adresa 255.255.255.255 „limited boradcast“ nesmí být přeposílána do internetu. Zbylé adresy jsou rezervovány pro budoucí použití RFC1700 (http://www.rfc-editor.org/rfc/rfc1700.txt) strana 4.

```
24.0.0.0/8
39.0.0.0/8
128.0.0.0/16
169.254.0.0/16
191.255.0.0/16
192.0.0.0/24
192.88.99.0/24
198.18.0.0/15
223.255.255.0/24
```

FIXME: popsat

24.2. Nastavení síťových rozhraní

24.2.1. ifconfig

24.2.2. /etc/network/interfaces

Ukázka manuální konfigurace rozhraní

```
iface eth0 inet static
        address 192.168.1.1
        netmask 255.255.255.0
```

Konfigurace rozhraní pro použití DHCP.

```
iface eth0 inet dhcp
```

24.3. Omezování datového toku

* *section*

Pro omezování datového toku je možno použít externí program shaper, nebo možnosti nového jádra (od verze 2.2 výše) jenž se jmenuje QoS.

24.3.1. Použití skriptu CBQ.init

* *section id="cbq.init" xreflabel="cbq.init"*

Skript `cbq.init` se nachází na adrese a http://uf.kadan.cz/cbq (https://sourceforge.net/projects/cbqinit)

Tedy musíme mít zakompilovánu v kernelu podporu pro QoS. Volba *Networking options / IP: advanced router*, a v jádře buď přímo, nebo jako moduly jednotlivé QoS moduly. Pak nainstalujeme iproute a do adresáře `/etc/init.d/` vložíme skript `cbq`.

```
# cp cbq.init-v0.7 /etc/init.d/cbq-init
# chmod u+x /etc/init.d/cbq-init
```

Nastavíme cestu k adresáři s konfiguracemi vložením následujícího řádku do souboru.

```
CBQ_PATH=/etc/cbq
```

Skript připravíme k použití příkazem **update-rc.d**.

```
# update-rc.d cbq defaults
```

Zbývá nám připravit konfiguraci pro CBQ. Ta sestává ze souborů v adresáři `/etc/cbq` který musíme také vytvořit.

Příklad 24-1. Ukázka omzení rychlosti na eth1

```
DEVICE=eth1,10Mbit,1Mbit
RATE=32Kbit
WEIGHT=3Kbit
PRIO=5
RULE=192.168.254.0/24
```

24.3.2. HTB.init

Zdroje a odkazy:

- HTB Linux queuing discipline manual - user guide (http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm)
- HTB Home (http://luxik.cdi.cz/~devik/qos/htb/)
- Traffic Shaping with Linux v2.4 and HTB qdisc (http://www.trekweb.com/~jasonb/articles/linux_tc_minihowto.shtml)

FIXME:

24.3.3. RShaper

* *section id="rshaper" xreflabel="RShaper"*

- Alessandro Rubini (http://arcana.linux.it/software/#rshaper)
- RShaper - omezení rychlosti síťového provozu (http://poweroff.cz/index.php?clanek=16)

RShaper je modul do jádra od Alessandra Rubiniho. Je velmi jednoduchý na instalaci i použití. Pro řadu jednodušších síťových konfigurací je zajímavý právě svou jednoduchostí.

K modulu do jádra je k dispozici ovládací program **rshapectl**.

24.3.3.1. Instalace

K překladu modulu budeme potřebovat hlavičkové soubory aktuálně bežícího jádra. Pokud jsem si překládali jádro sami, nastavíme proměnnou `KERNELDIR` a adresář se zdroji. Pokud ne, nainstalujeme hlavičkové sooubory. V ukázce používám jádro `2.4.18-686`

```
# apt-get install gcc glibc-dev kernel-headers-2.4.18-686
# tar xzf rshaper-2.01.tar.gz
# cd rshaper-2.01
# export KERNELDIR=/usr/src/kernel-headers-2.4.18-686
# make
# make install
# depmod -a
```

Po překladu můžeme modul zavést a odzkoušet.

```
# modprobe rshaper
```

Po odzkoušení upravíme konfiguraci aby se modul zaváděl při startu systému. Do souboru `/etc/modules` přidáme následující řádku.

```
rshaper
```

Do adresáře `/etc/modutils` vložíme soubor `rshaper` s obsahem

```
options rshaper mode=2
```

A přegenerujeme konfiguraci modulů

```
# update-modules
```

Poznámka: Parametr mode určuje mód v kterém **rshaper** pracuje. Bude vysvětlen dále.

24.3.3.2. Konfigurace

První věcí kterou si musíme určit je mód ve kterém bude **rshaper** pracovat. Tento se zadává jako číselný parameter *mode* a může nabývat hodnot

0 — filtrace odchozích paketů
1 — filtrace příchozích paketů
2 — obousměrná filtrace příchozích i odchozích paketů (vyžaduje jádro 2.4)

Další část konfigurace se provádí programem **rshaperctl**.

rshaperctl [-nt] [address[/mask] bytes-per-second [queue-len]]

24.3.4. HTB

```
#!/bin/sh -x

tc qdisc add dev eth0 root handle 1: htb default 20

tc class add dev eth0 parent 1: classid 1:1 htb rate 100mbit ceil 100mbit
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 99mbit ceil 100mbit
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 1mbit ceil 100mbit
tc class add dev eth0 parent 1:11 classid 1:20 htb rate 20kbps ceil 30kbps high prio internet
tc class add dev eth0 parent 1:11 classid 1:21 htb rate 5kbps ceil 30kbps #high prio class
tc class add dev eth0 parent 1:11 classid 1:22 htb rate 5kbps ceil 30kbps #mldonkey's class

iptables -A POSTROUTING -t mangle -o eth0 -p tcp -m length --length :64 -j MARK --set-mark 21

tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 21 fw flowid 1:21
tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 22 fw flowid 1:22
tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 10 fw flowid 1:10
```

24.4. Bridging

* *section*

Linux umí ve verzích jádra 2.2.x bridging

Postup:

1. Je potřeba zakompilovat do jádra podporu pro bridging.
2. Nainstalovat pro jádro 2.2.x balíček bridgex pro jádro 2.0.x pak bridge
3. Popravit soubor `/etc/init.d/bridgex`. Do proměnné `INTERFACES` přiřadíme seznam všech rozhraní propojených do bridge. Opravíme řádek 14 z původního

   ```
   for i in $INTERFACES; do ifconfig $i up promisc; done
   ```

   na

   ```
   for i in $INTERFACES; do ifconfig $i up promisc; brcfg device $i enable; done
   ```

   Stejným způsobem opravíme řádek 22 z původních

   ```
   for i in $INTERFACES; do ifconfig $i -promisc; done
   ```

   na

```
for i in $INTERFACES; do brcfg device $i disable; ifconfig $i -promisc; done
```

24.5. Pokročilé síťování s jádrem 2.2.x (ipchains)

* *section id="linux-2.2.x-networking"*

Programy:

- ip

Pro některé volby je pořeba mít zakompilovány příslušné komponenty v jádře. Například chybová hláška *RTNETLINK answers: Invalid argument* může být způsobena tím, že při konfigurování jádra nebyla aktivována volba *IP: policy routing* v sekci *Networking options*.

Programem ip si vypíšeme směrovací tabulku:

```
# ip route
192.168.199.0/30 dev eth0  proto kernel  scope link  src 192.168.199.2
192.168.199.4/30 via 192.168.199.1 dev eth0
10.16.64.0/19 dev eth1  proto kernel  scope link  src 10.16.66.51
10.32.64.0/19 via 192.168.199.6 dev tun0 onlink
10.48.0.0/13 via 10.16.66.60 dev eth1
10.32.0.0/13 via 192.168.199.1 dev eth0
```

24.5.1. Průzkum konfigurace

Nejprve prozkoumáme svoji aktuální konfiguraci.

Síťová rozhraní na počítači si zjistíme příkazem link list

```
pikara:~$ /sbin/ip link list
1: lo: <LOOPBACK,UP> mtu 3924 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: brg0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop
    link/ether fe:fd:00:00:15:b8 brd ff:ff:ff:ff:ff:ff
3: tunl0@NONE: <NOARP> mtu 1480 qdisc noop
    link/ipip 0.0.0.0 brd 0.0.0.0
4: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:03:47:3b:25:b0 brd ff:ff:ff:ff:ff:ff
5: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:03:47:3b:25:b1 brd ff:ff:ff:ff:ff:ff
6: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 100
    link/ether 00:d0:b7:00:ca:51 brd ff:ff:ff:ff:ff:ff
7: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 100
    link/ether 00:d0:b7:00:ca:52 brd ff:ff:ff:ff:ff:ff
8: tun0@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
    link/ipip 0.0.0.0 peer 192.168.199.6
```

Aktuální adresy rozhraní. zjistíme

```
pikara:~$ /sbin/ip address show
1: lo: <LOOPBACK,UP> mtu 3924 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
2: brg0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop
    link/ether fe:fd:00:00:15:b8 brd ff:ff:ff:ff:ff:ff
3: tunl0@NONE: <NOARP> mtu 1480 qdisc noop
```

```
    link/ipip 0.0.0.0 brd 0.0.0.0
4: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:03:47:3b:25:b0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.199.2/30 brd 192.168.199.3 scope global eth0
5: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:03:47:3b:25:b1 brd ff:ff:ff:ff:ff:ff
    inet 10.16.66.51/19 brd 10.16.127.255 scope global eth1
6: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 100
    link/ether 00:d0:b7:00:ca:51 brd ff:ff:ff:ff:ff:ff
7: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 100
    link/ether 00:d0:b7:00:ca:52 brd ff:ff:ff:ff:ff:ff
8: tun0@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
    link/ipip 0.0.0.0 peer 192.168.199.6
    inet 10.16.66.51/32 scope global tun0
```

Směrovací tabulku. nám pak ukáže

```
pikara:~$ /sbin/ip route show
192.168.199.0/30 dev eth0  proto kernel  scope link  src 192.168.199.2
192.168.199.4/30 via 192.168.199.1 dev eth0
10.16.64.0/19 dev eth1  proto kernel  scope link  src 10.16.66.51
10.32.64.0/19 via 192.168.199.6 dev tun0 onlink
10.48.0.0/13 via 10.16.66.60 dev eth1
10.32.0.0/13 via 192.168.199.1 dev eth0
```

ARP table. ARP/neighbor cache/table

```
pikara:~$ /sbin/ip neigh show
10.16.66.18 dev eth1 lladdr 00:90:27:f7:19:1e nud reachable
10.16.68.2 dev eth1 lladdr 00:b0:d0:16:21:12 nud reachable
192.168.199.1 dev eth0 lladdr 00:04:9a:41:0d:20 nud reachable
```

Směrovací tabulky.

```
pikara:~$ /sbin/ip rule list
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

24.5.2. Přesměrování provozu na portu jinam

```
# ipmasqadm portfw -a -P tcp -L 192.168.0.1 25 -R 192.168.0.2 25
```

24.6. Síťování s jádrem řady 2.4.x

* *section id="fragment.kernel_2.4.x_networking" condition="author"*

Zdroje a odkazy:

- http://soban.wz.cz/linux/firewall.html

- http://www.faqs.org/docs/Linux-mini/TransparentProxy.html

24.6.1. Průchod paketu tabulkami a řetezci

24.6.1.1. Tabulka `mangle`

V tabulce `mangle` je možno použít jen akce `TOS`, `TTL` a `MARK`

24.6.1.2. Tabulka `nat`

Používá se jen pro překlad adres (Network Address Translation).

Je možno použít jen akce `DNAT`, `SNAT` a `MASQUERADE`.

24.6.1.3. Tabulka `filter`

FIXME:

24.6.2. iptables

* *section id="fragment.iptables" condition="author"*

Zdroje a odkazy:

- http://www.fi.muni.cz/~xsafran1/linux/netfilter.htmlFirewall - paketový filtr

Ukázka přesměrování portu na hradebním počítači na jiný počítač uvnitř sítě.

```
# iptables -t nat -A PREROUTING -p tcp -d old_mail_server \
        --dport 25 -j DNAT -- new_mail_server:25
```

Zdroje a odkazy:

- netfilter (http://www.netfilter.org/)

24.6.2.1. Příprava a instalace

Před prvním použitím musíme nainstalovat nezbytné programy a případně přeložit jádro.

```
# apt-get install iptables
```

V jádro musí být přeloženo s volbami

`CONFIG_PACKET`

FIXME: doplnit

`CONFIG_NETFILTER`

FIXME: doplnit

`CONFIG_IP_NF_CONNTRACK` — *connection tracking*

FIXME: doplnit

`CONFIG_IP_NF_FTP` — *connection tracking on FTP*

FIXME: doplnit

`CONFIG_IP_NF_IPTABLES`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_LIMIT`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_MAC`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_MARK`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_MULTIPORT`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_TOS`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_TCPMSS`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_STATE`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_UNCLEAN`

FIXME: doplnit

`CONFIG_IP_NF_MATCH_OWNER`

FIXME: doplnit

`CONFIG_IP_NF_FILTER`

FIXME: doplnit

`CONFIG_IP_NF_TARGET_REJECT`

FIXME: doplnit

`CONFIG_IP_NF_TARGET_MIRROR`

FIXME: doplnit

`CONFIG_IP_NF_NAT`

FIXME: doplnit

`CONFIG_IP_NF_TARGET_MASQUERADE`

FIXME: doplnit

`CONFIG_IP_NF_TARGET_REDIRECT`

FIXME: doplnit

`CONFIG_IP_NF_TARGET_LOG`

FIXME: doplnit

`CONFIG_IP_NF_TARGET_TCPMSS`

FIXME: doplnit

`CONFIG_IP_NF_COMPAT_IPCHAINS`

FIXME: doplnit

`CONFIG_IP_NF_COMPAT_IPFWADM`

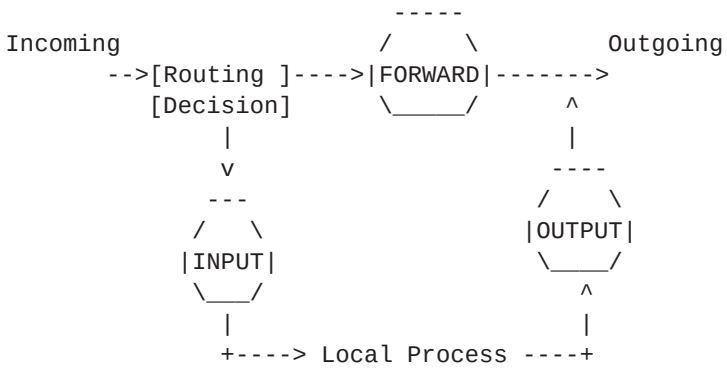
FIXME: doplnit

`CONFIG_NETFILTER  Yes`

24.6.2.2. Překlad iproute

```
# apt-get source iproute
# apt-get install gcc tetex-bin atm-dev
```

24.6.2.3. Jak procházejí pakety filtrem



24.6.2.4. Nastavováni paketového filtru

S paketovým filtrem komunikujeme pomocí programu iptables. Pomocí něj nastavujeme standardní policy a přidáváme a ubíráme pravidla. Pro přídávín a odebírání pravidel máme volby

Tabulka 24-2. Přidávání a odebírání pravidel s pomocí iptables

`-A`	přidání pravidla na konec
`-I`	přidání pravidla na začátek
`-D`	odstranění pravidla
`-R`	výměna pravidla

24.6.2.5. Popis shody

V této části si něco povíme o tom jak specifikovat pakety které nás zajímají.

24.6.2.5.1. MAC

Porovnávání MAC adresy

Příklad:

```
# iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01
```

24.6.2.6. Cíle / akce

Cíle, někdy zvané akce jsou parametry přepínače `-j`

24.6.2.6.1. ACCEPT

FIXME:

24.6.2.6.2. DNAT

FIXME:

24.6.2.6.3. DROP

FIXME:

24.6.2.6.4. LOG

FIXME:

24.6.2.6.5. MARK

FIXME:

24.6.2.6.6. MASQUERADE

FIXME:

24.6.2.6.7. MIRROR

FIXME:

24.6.2.6.8. QUEUE

FIXME:

24.6.2.6.9. REDIRECT

FIXME:

24.6.2.6.10. REJECT

FIXME:

24.6.2.6.11. RETURN

FIXME:

24.6.2.6.12. SNAT

FIXME:

24.6.2.6.13. TOS

FIXME:

24.6.2.6.14. TTL

FIXME:

24.6.2.6.15. ULOG

FIXME:

24.6.2.7. Ruční nastavení maškarády

Jednoduchý script pro nastavení maškarády. Je na něm vidět jak se používá program `iptables`

```
# Zavést NAT modul (to zavede všechny ostatní).
modprobe iptable_nat

# V tabulce NAT (-t nat) připoj pravidlo (-A), že po dokončení směrování
# (POSTROUTING) má být u všech paketů odcházejících z ppp0 (-o ppp0) prováděno
# MASQUERADE spojení (-j MASQUERADE).
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Zapnutí IP forwardování
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Další ukázky pravidel pro sestavení maškarády

```
# iptables -t nat -A POSTROUTING -s 192.168.111.0/24 -j MASQUERADE
```

Jiný způsob sestavení maškarády který jsem použil u jednoho kamaráda je

```
# Building / Destroying Masquerade
up /sbin/iptables  -t nat -A POSTROUTING -o eth0 -j SNAT --to 212.96.165.22
down /sbin/iptables  -t nat -D POSTROUTING -o eth0 -j SNAT --to 212.96.165.22
```

slova `up` a `down` označují v kterém okamžiku se daný příkaz vykoná a pravidlo zavede či zruší. Tento kód je ze souboru `/etc/network/interfaces` a je v části náležející vnějšímu rozhraní `eth0`.

Ukázka nastavení maškarády na routeru jen pro část adres. V následující ukázce je:

```
                    +-------------+
                eth0 |             | eth1
---------------------| R O U T E R |--------------------
             1.2.3.4 |             | 172.19.1.1/24
                    +-------------+
```

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE -s 172.19.1.0/24
```

24.6.2.8. Řešení vybraných problémů

Odkazy:

- iptables-1.2.4-2 (http://www.grape-info.com/doc/linux/config/iptables-1.2.3-1.html)
- Configuring IP Masquerade on Linux 2.6.x and 2.4.x Kernels (http://www.tldp.org/HOWTO/IP-Masquerade-HOWTO/firewall-examples.html#RC.FIREWALL-IPTABLES)
- **FIXME:** ()

Zde popisuju ukázková řešení některých problémů s nimiž se při konfiguraci firewallu potkáváme.

Použité konvence

- `eth0` je vždy venkovní rozhraní, tedy rozhraní připojené do internetu a `eth1` a vyšší pak vnitřní rozhraní, tedy rozhraní připojené k lokální síti.

24.6.2.8.1. Maškaráda

Maskování provozu celé sítě počitaču na neveřejných adresách za veřejnou adresu routeru.

Před nastavením samotné maškarády nastavíme default policy pro vstupní a výstupní pakety na ACCEPT a pro forwardované pakety na DROP. Rovněž vyprázdníme všechny tabulky.

```
# iptables -P INPUT ACCEPT
# iptables -F INPUT
# iptables -P OUTPUT ACCEPT
# iptables -F OUTPUT
# iptables -P FORWARD DROP
# iptables -F FORWARD
# iptables -t nat -F
```

Poté povolíme forward paketů z venku (INTERNET) dovnitř (LAN) pro již navázaná spojení a RELATED spojení. Povolíme také forward všech paketů zevnitř (LAN) ven (INTERNET). Jako poslední pravidlo do FORWARD řetězce přidáme "logovací" pravidlo jenž nám do deníku zaznamená všechny paket které byly odmítnuty.

```
# iptables -A FORWARD -i eth0 -i eth1 -m state --state ESTABLISHED,RELATED \
          -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
# iptables -A FORWARD -j LOG
```

A nakonec nám zůstane nastavení samotné maškarády.

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Pokud nechcem povolit "průchod" maškarádou všem počítačům na lokální síti ale jen některým, omezíme pomocí přepínače `-s` adresy na které se maškeráda vztahuje. Předpokládejme že vnitřní lan je síť s parametry 192.168.1.0/24. Povolení maškarády jen pro počítač s ip 192.168.1.3 a pro počítače s adresami od 192.168.1.64 do 192.168.1.95 provedeme příkazy.

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE -s 192.168.1.3
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE -s 192.168.1.64/27

# iptables -A INPUT -i eth0 -m state --state NEW,INVALID -j DROP
# iptables -A FORWARD -i eth0 -m state --state NEW,INVALID -j DROP
```

Více přísnější varianta maškarády s užitím SNAT. Adresa 1.2.3.4 v tomto příkladu je adresa vnějšího (INTERNET) rozhraní.

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
```

24.6.2.8.2. Transparent Proxy

Transparentní přesměrování veškerého tcp provozu na portu 80, tedy standardního webu procházejícího routerem na lokální port 8080 na kterém pak naslouchá proxyserver.

```
# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 \
          -j REDIRECT --to-port 8080
```

Pokud je třeba směrovat na jiný stroj, provedeme to takto. Provoz směrovaný původně na 1.2.3.4 port 8080, přesměrováváme na 192.168.1.1 port 80.

```
# iptables -t nat -A PREROUTING -p tcp -d 1.2.3.4 --dport 8080 \
          -j DNAT --to 192.168.1.1:80

# Transparent Proxy (to a Remote Box)
# iptables -t nat -A PREROUTING -i $IN_ETH -s ! $LOCAL_IP -p tcp \
#       --dport 80 -j DNAT --to $PROXY_IP:$PROXY_PORT
# iptables -t nat -A POSTROUTING -o eth0 -s $LOCAL_NET -d $PROXY_IP \
#       -j SNAT --to $LOCAL_IP
# iptables -A FORWARD -s $LOCAL_NET -d $PROXY_IP -i $IN_ETH -o $EX_ETH \
#       -p tcp --dport $PROXY_PORT -j ACCEPT
```

24.6.2.9. Nezařazeno

* *section condition="author"*

24.6.2.9.1. Poznámky a různé ukázky použití

Jak se podívat do pravidel:

```
# iptables -nvL
```

Přepínač `-L` bez argumentů (musí být poslední) zajistí vylistování pravidel ze všech řetězů (chains). Přepínač `-v` rozšíří výpis a `-n` zajistí že ip a porty budou vypsány čísly.

Nastavení policy:

```
# iptables -P forward DROP
```

24.6.2.9.1.1. Ukázka nastavení firewallu na počítači `yoda`

```
yoda:~# iptables -P FORWARD DROP
yoda:~# iptables -P INPUT DROP
yoda:~# iptables -A INPUT -j ACCEPT -s 10.16.66.18
yoda:~# iptables -A INPUT -j ACCEPT -s 10.16.68.2
```

24.6.2.9.2. Implicitní pravidla, policy

Implicitní pravidla, neboli policy se uplatní, když není vybráno žádné jiné pravidlo. Nastavuji se

```
# iptables -P fronta akce
```

Jako akce jsou dovoleny jen `ACCEPT` a `DENY`

24.6.2.9.3. Akce

Tabulka 24-3. Akce

akce	popis
`ACCEPT`	Let packet through
`DENY`	Odmítnutí paketu
`REJECT`	Odmítnutí paketu a upozornění odesílatele
`MASQ`	Maškaráda (pro frontu `FORWARD`)
`REDIRECT`	Odeslání na odlišný port
`RETURN`	Handled by default targets

24.6.2.10. Nezapracované poznámky

Jinak není potřeba zakazova ECN úplně, dá se použít

```
iptables ... -j ECN --ecn-tcp-remove
```

(v tabulce mangle) jen pro problémové cíle.

24.6.3. Firewall

* *$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $*
* *section id="fragment.firewall"*

Zdroje a odkazy:

- ftester (http://ftester.sourceforge.net/), SourceForge summary page (http://sourceforge.net/projects/ftester/) — Testování firewallu

ToDo list

1. FIXME:

FIXME:

24.6.3.1. Shorewall (Shoreline Firewall)

* *$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $*
* *section id="fragment.shorewall"*

Zdroje a odkazy:

- FIXME:

ToDo list

1. FIXME:

FIXME:

Shorewall je firewall založený na netfiltru (*netfilter*, *iptables*).

Konfigurační soubory

- `/etc/shorewall/zones`

  Svět je pro `shorewall` rozdělen do zón. `shorewall` sám patří do zóny `fw`

- `/etc/shorewall/policy`

  Defaultní pravidla pro připojení z jednotlivých zón.

- `/etc/shorewall/rules`

  Výjimky z defaultních pravidel.

24.6.3.1.1. Instalace

Ve Woody je k dispozici verze `1.2.12-1`. Pokud chceme novější, můžeme si nainstalovat `1.4.5-1` ze Sarge.

Instalace je jednoduchá

```
# apt-get install shorewall
```

a můžeme si nainstalovat i dokumentaci

```
# apt-get install shorewall-doc
```

24.6.3.1.2. Testování konfigurace

Shorewall má v sobě zabudovánu podporu pro testování konfigurace.

Při spuštění s parametrem `try` se budou konfigurační soubory přednostně (jestli jsou) číst z uvedeného adresáře a po stanoveném čase se uvede firewall do původního stavu.

```
# shorewall try adresář_s_konfigurací [timeout]
```

24.6.3.2. Použití programu netstat

* *section id="program.netstat" condition="author"*

Program **netstat** slouží k zjišťování informací o aktuálně otevřených spojeních, obsazených portech a programech jenž je používají.

24.6.3.2.1. Nezapracované informace

Ukázky použití

```
# netstat -ain
```

24.6.3.3. LARTC — Linux Advanced Routing & Traffic Control

Zdroje a odkazy:

- Linux Advanced Routing & Traffic Control (http://lartc.org/)

FIXME:

Kapitola 25. DNS *Domain Name System*

* *chapter id="dns"*

Adresy počítačů v sítích TCP/IP jsou:

- 32 bitová čísla, zapisovaná jako čtyři oktety v případě IPv4
- 128 bitová čísla, zapisovaná jako 8 skupin v případě IPv6

Přistupování k počítačům pomocí IP adresy je tedy pro člověka ne zrovna nejpříjemnější. Taková čísla se nedobře zapamatovávají. Proto byl v počátcích internetu

* ***FIXME:****specifikovat dobu.*

vytvořen systém symbolických jmen. Taková jména se pamatují a používají lidem mnohem lépe. Systém jmen je hierarchický a vychází ze stromové struktury.

Pro sprovoznění domény druhého řádu v doméně `cz` je třeba mimo jiné povolit přenos zónového souboru pro počítače NICu. Jedná se o tyto počítače

Tabulka 25-1. Nameservery NIC CZ

server	ip adresa
ns.o.cz	193.85.7.100
ns.eunet.cz	193.85.1.12
cz.eunet.cz	193.85.3.130

Programy a balíčky k prozkoumání

- dns-browse — `dns-browse` — Frontends to DNS search

25.1. Jak DNS funguje a k čemu je?

DNS neboli systém systém doménových jmen internetu (*Internet Domain Name System*)

Seznam kořenových serverů musí být aktuální. V případě

```
# ps ax|grep named| grep -v grep
# netstat -a |grep domain
# nslookup
```

25.2. Testování konfigurace DNS serverů

Servery na internetu testující DNS:

- Uniplace CheckDNS (http://www.checkdns.net)
- The DNS Sleuth (http://atrey.karlin.mff.cuni.cz/~mj/sleuth/)
- The DNS Sleuth (http://opal.unipo.sk/cgi-bin/sleuth/check1)

25.3. bind

* *section id="bind" xreflabel="bind"*

Odkazy a zdroje:

- **FIXME:**OpenBSD 6.0 - Nastavení sítě (http://openbsd.shellhung.org/faq/cs/faq6.html)
- ISC BIND (http://www.isc.org/index.pl?/sw/bind/)
- **FIXME:**Open Source Guide about DNS (http://www.zytrax.com/books/dns/)

BIND, neboli *Berkeley Internet Name Domain* je původní internetový program pro rezoluci symbolických doménových jmen na ip adresy.

Po změně konfigurace binda, přesněji po úpravě zónového souboru je potřeba říci bindovi, aby si nahrál změny. To učiníme buď

```
# ndc restart
```

nebo

```
# kill -HUP <pid_of_nameserver>
```

V Debianu stačí

```
# /etc/init.d/bind/reload
```

25.3.1. Zónové soubory

* *section id="bind.zone-file"*

Nejdříve malou ukázku, kterou si dále rozvedeme.

```
$ORIGIN moje-firma.cz.
$TTL    1D
@       IN      SOA     moje-firma.cz.     tomas.muj-mail.cz. (
                        20050116                ; Serial
                        86400                   ; Refresh       24h
                        3600                    ; Retry         1h
                        2419200                 ; Expire        28d
                        604800                  ; Default TTL   7d
                        ;;; High speed actualization
                        ;14400                  ; Refresh       4h
                        ;3600                   ; Retry         1h
                        ;172800                 ; Expire        2d
                        ;86400                  ; Default TTL   1d
                )

                IN      A       312.296.165.116          ;horace.moje-firma.cz

                IN      NS      horace.moje-firma.cz.
                IN      NS      @

                IN      MX      10      horace.moje-firma.cz.
                IN      MX      50      jilji.jina-firma.cz.

horace          IN      A       312.296.165.116
grok            IN      A       312.296.165.117

mail            IN      CNAME   horace.moje-firma.cz.
```

```
ftp             IN      CNAME   erol.jina-firma.cz.
www             IN      CNAME   horace.moje-firma.cz.
cvs             IN      CNAME   horace.moje-firma.cz.

; Sekundární WWW server
www2            IN      CNAME   wufi.jina-firma.cz.
```

> **Varování**
>
> Nezkoušejte uvedený soubor použít. Některé hodnoty jsou úmyslně nesmyslné a řada jmen, z reálně používaného původního souboru, byla přepsána bez další kontroly. Nemusí proto dávat smysl.

25.3.1.1. Zónový soubor pro delegaci v reverzní doméně

```
; Pro delegaci reverzni domeny
$TTL    3600
$ORIGIN 112/28.115.296.312.in-addr.arpa. ; added for clarity
112/28.115.296.312.in-addr.arpa. IN      SOA     erol.firma.cz.   radek.firma.cz. (
                        20041020        ; Serial number
                        86400           ; Refresh       24h
                        3600            ; Retry         1h
                        2419200         ; Expire        28d
                        604800                  ; Default TTL   7d
                        ;;; High speed actualization
                        ;14400                  ; Refresh       4h
                        ;3600                   ; Retry         1h
                        ;172800                 ; Expire        2d
                        ;86400                  ; Default TTL   1d
                )
        IN      NS      moraviapress.cz.

;112 - network address 312.296.115.112/28
113             IN      PTR     firma.cz.
113             IN      PTR     erol.firma.cz.
114             IN      PTR     grok.firma.cz.
115             IN      PTR     lutmi.firma.cz.
116             IN      PTR     horace.jinafirma.cz.
117             IN      PTR     117.firma.cz.
118             IN      PTR     118.firma.cz.
119             IN      PTR     119.firma.cz.
120             IN      PTR     120.firma.cz.
121             IN      PTR     121.firma.cz.
122             IN      PTR     122.firma.cz.
123             IN      PTR     deneb.firma.cz.
124             IN      PTR     pikara.firma.cz.
125             IN      PTR     ilja.firma.cz.
126             IN      PTR     kertok.firma.cz.
;127 - broadcast address
```

25.3.1.2. Typy záznamů v zónovém souboru.

Zdroje a odkazy:

- The Concise Guide to DNS and BIND (http://www.informit.com/title/0789722739)

A

FIXME:záznam adresy (IPv4 adresy)

AAAA

FIXME:záznam adresy (IPv6 adresy)

ATMA

FIXME:záznam ATM adresy

CNAME

FIXME:*Canonical Name of an Alias*

HINFO

FIXME:*Host Information*

MX

FIXME:*Mail Exchanger*

NS

FIXME:*Authoritative Nameserver*

NSAP ??

FIXME:

PTR

FIXME:*Pointer to Other Name*

Používá se v definici reverzní domény. Přiřazuje symbolické jméno ip adrese.

```
10          IN  PTR  chef.tech-recipes.com.
```

Nebo z jiné domény

```
126             IN      PTR     kertok.jina-firma.cz.
```

PX

FIXME:*X.400 Mapping*

RP

FIXME:*Responsible Person*

RT

FIXME:*Route Through*

SOA

FIXME:*Start Of Authority*

SRV

FIXME:*Service Locator*

TXT

FIXME:*Text Information*

X25

FIXME:*X25 Routing Information*

AFSDB

FIXME:*AFS Database Location*

ISDN

FIXME:.

KEY

FIXME:*Public Key*

LOC

FIXME:*Location*

KX

FIXME:*Key Exchange*

NULL

FIXME:.

NAPTR

FIXME:*Name Authority Pointer*

Další typy záznamů u nichž jsem našel zmínku a dále je nezkoumal.

- `NXT` — **FIXME:***Next Valid Name* (The Concise Guide to DNS and BIND (http://www.informit.com/title/0789722739))
- `SIG` — **FIXME:***Signature* (The Concise Guide to DNS and BIND (http://www.informit.com/title/0789722739))
- `SINK` — **FIXME:***The Kitchen Sink record* (The Concise Guide to DNS and BIND (http://www.informit.com/title/0789722739))
- `DNAME` —**FIXME:**RFC2672

Proměnné použitelné v zónovém souboru

$ORIGIN
@

FIXME:

$TTL

FIXME:

$INCLUDE

FIXME:

```
$INCLUDE /usr/etc/named.d/mailboxes
```

25.3.2. Ukázková instalace sekundárního serveru v Debian Woody

Instalace na hlavním serveru v satelitní lokaci. Tento slouží jako sekundární a odkazuje se na firmemní autoritativní servery `sunrise` (`10.16.66.18`, `dns`, a `ferit` (`10.16.66.19`). Jako software používám bind9

```
# apt-get install bind9
```

v souboru `/etc/bind/named.conf` v části `options` uvádím

```
        forwarders {
                10.17.66.28;    //Authoritative primary DNS sunrise
                10.17.66.29;    //Authoritative primary DNS ferit
        };
```

a definuji autoritativní servery pro firemní doménu `firma.cz`

```
// add entries for other zones below here
zone "firma.cz" {
        type slave;
        masters {
                10.17.66.28;    // Authoritative primary DNS sunrise
                10.17.66.29;    // Authoritative primary DNS ferit
        };
};
```

Ještě nastavím konfiguraci *resolveru* v souboru `/etc/resolv.conf`

```
search firma.cz
nameserver 127.0.0.1    # na lokale bezi sekundarni DNS
nameserver 10.17.66.28  # authoritative primary DNS sunrise
nameserver 10.17.66.29  # authoritative primary DNS ferit
```

25.3.3. Použití Webmin ke konfiguraci DNS

Nejdříve vytvořit reverzní zónu.

Servers / BIND DNS Server / Create Master Zone
o Revers (Address to Names)

Poté stejným způsobem vytvoříme doménu.

25.4. bind9

Bind verze 9 je novější verze programu bind.

25.4.1. Jak provozovat bind9 v chroot prostředí

```
# adduser --system --home /var/cache/bind --gecos 'BIND Server' --group --disabled-password bi
Adding system user bind...
Adding new group bind (101).
Adding new user bind (101) with group bind.
Creating home directory /var/cache/bind.
# cd /etc/init.d
# mv bind9 bind9.orig
# wget http://www.cryptio.net/~ferlatte/config/bind9.init
# mv bind9.init bind9
# chmod a+x bind9
# mkdir -p /chroot/bind
# vi /etc/default/bind9
```

Upravit konfiguraci a opravit chyby

25.5. La MaraDNS

La MaraDNS (http://www.maradns.org) je alternativou k bindu, který je obecně považován za ne zcela bezpečný. V Debianu se nachází od verze Potato.

* *FIXME: Nastudovat, odzkoušet, popsat*

25.6. djbdns

Další alternatiou pro poskytování služeb DNS je program, přesněji řečeno skupina programů s názvem djbdns (http://cr.yp.to/djbdns.html)

25.7. MyDNS

Odkazy, zdroje:

- MyDNS (http://mydns.bboy.net/)

Primární DNS server jenž má data uložena v databázi MySQL.

25.8. Registrace domén v `.cz`

Zde popisuji jak zaregistrovat doménu druhého řádu v doméně `.cz`.

U NICu (http://www.nic.cz/) je k dispozici seznam registrátorů. (http://www.nic.cz/page.php?sid=9) Tady si pár z nich uvedu.

25.8.1. Czech On Line (http://domeny.col.cz)

Zdá se, že tento registrátor má Nástroje - pro pokročilé uživatele (http://domeny.col.cz/nastroje.php)

25.8.2. Domain Master (http://www.domainmaster.cz/)

FIXME: prozkoumat

25.8.3. DOMÉNY.CZ (http://www.domeny.cz/)

FIXME:

25.8.4. ha-vel (http://domeny.ha-vel.cz)

FIXME:

25.8.5. Domena.cz (http://www.domena.cz)

FIXME:

25.8.6. domeny.velkoobchod.cz (http://domeny.velkoobchod.cz/)

FIXME:

> **Varování**
>
> Poplatek za používání cizích DNS serverů je 50 Kč ročně.

25.8.7. 9net.cz (http://www.9net.cz)

FIXME:

25.8.8. RegZone! (http://www.regzone.cz)

FIXME:

25.8.9. Internet OnLine (http://domeny.iol.cz)

FIXME:

25.8.10. XNET (http://www.xnet.cz)

FIXME:

25.8.11. Media4web (http://www.media4web.cz)

FIXME: prozkoumat

25.8.12. REGISTRATOR.CZ (http://www.registrator.cz)

FIXME:

25.8.13. Správa domén (http://www.spravadomen.cz)

FIXME:

25.8.14. NEXTRA (http://domeny.nextra.cz)

FIXME:

25.8.15. OK domény (InWay, a.s.) (http://domeny.ok.cz)

Obchodní oddělení:tel: +420 226 204 111, e-mail: obchod@inway.cz

Helpdesk 24 hodin denně:tel: +420 226 204 400, e-mail: helpdesk@inway.cz

FIXME: prozkoumat

25.8.16. GIN (http://domeny.gin.cz)

Nic neříkající prázdný web.

25.8.17. SkyNet (http://www.skynet.cz)

Tady jsem zmaten, nechápu proč je SkyNet uváděn NICem jako registrátor, když jsem o tom na webu na který NIC odkazuje nenašel nic.

25.8.18. LRR (http://www.lrr.cz)

FIXME:

25.9. Testování a vyhledávání chyb v DNS

Testování funkce systému provádíme programy **host** a **dig**. Někde se vyskytuje starší program **nslookup**.

```
yoda:~# apt-get install dnsutils
```

Kapitola 26. Ladění, sledování a testování sítě

Odkazy:
- Monitoring and Logging (http://www.sr.bham.ac.uk/~mpc/p2/monitor/)

26.1. Nástroje

26.1.1. tcpdump
dump a trafic on a network

Program tcpdump vypisuje provoz na síťovém rozhraní. Podle nastavení vypisuje jen hlavičky či celé pakety.

Například sledování provozu na rozhraní *eth0* provedeme příkazem

```
# tcpdump -i eth0
```

26.1.1.1. Příklady použití

Výpis komunikace s konkrétním počítačem:

```
# tcpdump -i eth0 -l host hostname
```

26.1.2. nmap
Aktivní skoumání počítačů

Program **nmap** slouží ke zkoumání síťového připojení počítačů. Zkouší které porty jsou na kterých počítačích přístupny a tím nás informuje o pravděpodobných službách jenž různé počítače poskytují síti.

Ukázkový příklad zkoumání které tiskárny máme na sítí, či počítače poskytující tiskové služby.

```
# nmap -p 515 192.168.10.*
```

26.1.3. dig
Zkoumání DNS

Potřebujeme-li zjistit jaké že to stroje jsou v dané doméně, můžemepoužít program dig

```
# dig -x 209.102.25.210 @b.root-servers.net
```

26.2. Měření průtoku dat

Sledování objemu protečených dat

Použitím balíčku ipac je možné nechat si vytvářet grafy zobrazující objem protečených dat jednotlivými rozhraními. Potřebujeme k tomu

- WWW server — stačí dhttpd
- Balíček ipac
- GD knihovnu pro Perl, která je v balíčku libgd-perl

Nainstalujeme WWW server, zkontrolujeme konfiguraci a vytvoříme adresáře

```
# wajig install dhttpd
# mkdir /var/www/ipac
# mkdir /var/www/ipac/4H
# mkdir /var/www/ipac/1D
# mkdir /var/www/ipac/1W
# mkdir /var/www/ipac/1M
# mkdir /var/www/ipac/1Y
```

nebo

```
# wajig install dhttpd
# mkdir /var/www/ipac
# mkdir /var/www/ipac/day
# mkdir /var/www/ipac/week
# mkdir /var/www/ipac/month
# mkdir /var/www/ipac/year
```

Tím máme připraven WWW server. Nyní nainstalujeme ipac a libgd-perl.

```
# wajig install ipac libgd-perl
```

Poté program `ipac` nakonfigurujeme. Můžeme vyjít z ukázkového souboru.

Příklad 26-1. Ukázkový soubor pro ipac

```
# /etc/ipac.conf
Port0|both|eth0|all||
Port1|both|eth1|all||
Port2|both|eth2|all||

Port0.in|in|eth0|all||
Port0.out|out|eth0|all||
Port1.in|in|eth1|all||
Port1.out|out|eth1|all||
Port2.in|in|eth2|all||
Port2.out|out|eth2|all||
```

V konfiguraci je možno *vypíchnout* konkrétní druh provozu, např http

```
http in (surf)|in|eth0|tcp|0/0 80|
```

nebo dns

```
dns in/out udp|both|eth0|udp||0/0 domain
```

```
dns in/out tcp|both|eth0|tcp||0/0 domain
```

či smtp

```
mail smtp in/out|both|eth0|tcp||0.0.0.0/0 smtp
```

Vše je připraveno, a ipac začal „sbírat“ data o průtoku. Zbývá nám přípravit generování grafů. Generování jsem vepsal do souboru `/etc/cron.d/ipac`. Sběr dat, řádek

```
*/10 * * * *  root  test -f /etc/ipac.conf && test -f /usr/sbin/fetchipac && test -f /var/run/
```

jsem ponechal na desetiminutových intervalech. Po dobu zkoušení je možno sbírat data rychleji, zkoušel jsem to po minutách. Protože 10-ti minutové intervaly jseou pro generování ročního a měsíčního grafu zcela zbytečně jemné, tak po cca 60 hodinách sčítam vše po hodinách.

```
0   *  *   *   *  root /usr/sbin/ipacsum -r -s 61h59m55s -e 60h59m55s >/dev/null
```

Tak a teď už jen zbýva generovat grafy:

```
*/10 * * * *  root  /usr/sbin/ipacsum -s 3h59m55s --gif /var/www/ipac/4H --gif-normalize 8 --g
*/10 * * * *  root  /usr/sbin/ipacsum -s 24h59m55s --gif /var/www/ipac/1D  --gif-normalize 8 -
0 * * * *  root  /usr/sbin/ipacsum -s 6D23h59m55s --gif /var/www/ipac/1W --gif-normalize 8 --g
0 0 * * *  root  /usr/sbin/ipacsum -s 30D23h59m55s --gif /var/www/ipac/1M --gif-normalize 8 --
0 0 * * *  root  /usr/sbin/ipacsum -s 365D23h59m55s --gif /var/www/ipac/1Y --gif-normalize 8 -
```

26.2.1. `ipac-ng`

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Program `ipac` funguje jen na jádrech řady 2.0 a 2.2. Na jádrech řady 2.4 již nefunguje. Pro tato jádra je určen program `ipac-ng`.

26.3. cricket - komplexní přehled o síti a tocích

Na tento nástroj jsem přišel při hledání programu pro měření kvality linek. Sice přímo kvalitu neměří, ale našel jsem k němu skript jenž tuto vlastnost přidává. Po zkoumání konfiurace se mi ho podařilo sprovoznit.

26.4. DNS - kontrola a „studium“

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Jak kontrolovat funkčnost DNS, procházet záznamy, ...

26.4.1. host

Program host slouží k základním informacím o doméně. Dovoluje položit dotaz na doménové jméno, či na reverzní dotaz libovolnému DNS serveru.

Pro zjištění IP adresy stroje `počítač`:

```
# host počítač
```

Pokud se potřebujeme zeptat jiného DNS serveru než který máme v konfiguraci (`/etc/resolv.conf`) použijeme:

```
# host počítač DNS_server
```

Dotaz na jméno počítače, známe-li jeho IP provedeme stejně, jen místo počítače napíšeme jeho IP:

```
# host IP DNS_server
```

26.5. Nagios

26.5.1. Instalace Nagios 1.2 z Backports.org do Woody

```
# apt-get install nagios-text
```

```
# su - postgres
# createdb nagios
# psql nagios
# create user nagios;
```

26.5.2. Poznámky k instalaci a používání Nagios3

Odkazy:

- Nagios 3.x Documentation (http://nagios.sourceforge.net/docs/3_0/toc.html)
- NagVis is a visualization addon for the well known network managment system Nagios. (http://www.nagvis.org/)

```
# aptitude search nagios|grep ^i
i A nagios-plugins-basic           - Plugins for the nagios network monitoring
i   nagios3                        - A host/service/network monitoring and mana
i A nagios3-common                 - support files for nagios3
i A nagios3-doc                    - documentation for nagios3
```

Příklad 26-2. `/etc/default/nagios3`

```
# /etc/default/nagios3
NAGIOSCFG="/etc/nagios3/nagios.cfg"
CGICFG="/etc/nagios3/cgi.cfg"
NICENESS=5
```

```
# dpkg-reconfigure nagios3-common
```

Vybral jsem:

- Apache servers: apache2
- Enable support for nagios 1.x links in nagios3: No
- Nagios web administration password: *heslo uživatele nagiosadmin*

Příklad 26-3. Ukázky vytržené z konfigurací (jyxo (http://usenet.jyxo.cz/cz.comp.linux/0706/nagios.html))

```
define host{
        use             generic-host
        host_name       Gateway-bright
        alias           Gateway-bright
        address         10.0.0.1
        parents         Gateway
}

define host{
        use             generic-host
        host_name       Dell01
        alias           Dell01
        address         10.0.0.2
        parents         Gateway-bright
}
```

Před nahráním nové konfigurace je vhodné ji zkontrolovat. Například následujícím příkazem.

```
# nagios3 -v /etc/nagios3/nagios.cfg
```

Příklad 26-4. Definice služby

```
define service{
        host_name               host_1, host_2, host_3, ..., host_n
        service_description     NějakáSlužba
        další direktivy
}
```

26.5.2.1. Konfigurace

26.5.2.1.1. host

```
define host{
        host_name               host_name       # required
        alias                   alias           # required
        display_name            display_name
```

```
⋮
}
```

26.6. RRD Tool

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Odkazy:

- RRDtool (http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/)
- . ()
- . ()

If you know MRTG, you can think of RRDtool as a reimplementation of MRTGs graphing and logging features. Magnitudes faster and more flexible than you ever thought possible

RRDtool je nástroj od Tobi Oetikera jenž realizuje RRD databázi a vytváření grafů z této databáze.

26.6.1. Instalace

K denšnímu dni (2005-03-15) jsou v Debianu balíčky:

- rrdtool 1.0.35-2, woody(stable)
- rrdtool 1.0.49-1, sarge(testing)

```
server:~# apt-get install rrdtool
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  libgd-gif1 libpng2 librrd0
The following NEW packages will be installed:
  libgd-gif1 libpng2 librrd0 rrdtool
0 packages upgraded, 4 newly installed, 0 to remove and 0  not upgraded.
Need to get 812kB of archives. After unpacking 2526kB will be used.
```

A to je všechno, žádná další konfigurace se neprovádí. RRDtool je nástroj podobně jako třeba … **FIXME:**

26.6.2. Hrajeme si

FIXME:

```
# rrdtool create target.rrd --start 1023654125 --step 300 \
          DS:mem:GAUGE:600:0:671744
          RRA:AVERAGE:0.5:12:24
          RRA:AVERAGE:0.5:288:31
```

26.7. Automatické sledování stavu sítě

Nástroje uvedené zde se používají hlavně k automatizovanému sledování sítě. Takový nástroj nám pak nějakým informačním kanálem oznamuje změny stavu sítě. Hlavní důvod pro jejich nasazení je informace o výpadcích služeb.

Kapitola 27. Virtuální privátní sítě (VPN)

Šablona pro nové kapitoly

Odkazy a zdroje:

- Odkaz ()

Virtuální privátní sítě (VPN) jsou jak již název napovídá virtuální sítě nad stávající síťovou technologií. Je to zposob jak propojit několik počítačů tak, aby měli bezpečné spojení mezi sebou a nenarušitelné či neodposlouchávatelné jinými do VPN nezahrnutými stroji.

Snad nejběžnějšími příklady VPN jsou:

- propojení sítí v pobočkách firmy s centrálou
- připojení mobilního pracovníka firemní sítě
- **FIXME:**doplnit další příklady

FIXME: Tato sekce pojednává o virtuálních sítích a tunelech jenžzajišťují spojení počítačů či sítí skrze jiné sítě a to většinou zabezpečeně. Například propojení sítí v pobočkách firmy tak, že se chovají jako jednotná vnitrofiremní síť.

27.1. CIPE

CIPE, neboli *Crypto IP Encapsulation* je rozšíření jádra Linuxu ve fromě modulu. Slouží k šifrování IP paketů a vytváření šifrovaných spojení mezi počítači.

Svého času jsem tento software hodně využíval, protože jeho nasazení bylo velmi jednoduché. Od roku 2005 není patrná žádná další vývojová aktivita. Tak jak to vidím se tento software dále nevyvýjí.

Zdroje a odkazy:

- CIPE - Crypto IP Encapsulation (http://sites.inka.de/sites/bigred/devel/cipe.html)
- VPN on Clark Connect (http://www.extra300.nl/cipe.htm)
- The Linux Cipe+Masquerading mini-HOWTO (http://www.tldp.org/HOWTO/Cipe+Masq.html)
- CIPE based VPNs (http://www.netpilot.com/products/CIPE/default.asp)
- Setting up VPN using Cipe (Crpto IP Encapsulation) (http://mia.ece.uic.edu/~papers/volans/cipe.html)
- Index of /security/cryptography/network/cipe (http://www.mirrors.wiretapped.net/security/cryptography/network/cipe/)
- Linux's answer to MS-PPTP (http://www.mit.edu:8008/bloom-picayune/crypto/14238)
- Jake Appelbaum summarizes flaws in "CIPE" Linux security tool (http://www.politechbot.com/pipermail/politech/2003-September/000038.html)
- CIPE (http://www.infoanarchy.org/wiki/wiki.pl?CIPE)
- Stavíme VPN - CIPE (http://www.root.cz/clanek/1307)
- Setting up and using NFS over CIPE tunnels (http://they.gotdns.org:88/~tscanlan/linux/nfs-cipe.php)
- Red Hat Linux 8.0: The Official Red Hat Linux Security Guide (http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/s1-vpn-cipe.html)
- CIPE An IP encription package (http://homepage.smc.edu/morgan_david/vpn/cipe-1_5_1_toc.html)
- CIPE - Cryptographic IP Encapsulation (http://homepage.smc.edu/morgan_david/vpn/cipe.htm)

- Red Hat Linux 9: Red Hat Linux Security Guide (http://docs.biostat.wustl.edu/rhl-sg-en-9/s1-vpn-cipe-server.html)
- My response to both the analysis of CIPE by Gutmann, Slashdot and the response by the CIPE list (http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2003-09/0418.html)
- Setting up VPNs easily with CIPE (http://www.pycs.net/lateral/stories/5.html)
- CIPE (http://www.merit.ee/~gunnar/)
- CIPE protocol for 20% faster VPN (http://www.netpremacy.com/cipe.htm)
- [security] CIPE (http://lists.linux.sk/pipermail/security/2000-November/001486.html)
- Re: Compliling Cipe for Debian 3.0r1 (http://sites.inka.de/bigred/archive/cipe-l/2003-08/msg00080.html)

Další z možností jak realizovat virtuální privátní síť je program CIPE

Uvedu příklad konfigurace. Potřebuji propojit dvě sítě. Obě jsou přes vlastní směrovač připojeny k internetu.

```
ipaddr  192.168.65.1
ptpaddr 192.168.65.2

me      217.66.164.3:6789
peer    217.66.164.7:6789

device  cipcb0
maxerr  -1

key     tajný klíč shodný pro obě strany

ipaddr  192.168.65.2
ptpaddr 192.168.65.1

me      217.66.164.7:6789
peer    217.66.164.3:6789

device  cipcb0
maxerr  -1

key     tajný klíč shodný pro obě strany
```

27.1.1. Překlad a instalace

Protože si pro vlastní potřebu vždycky překládám jádro sám, nemohu využívat řady přeložených balíčků jenž mají úzkou vazbu právě na jádro. Z tohoto důvodu jsem si taky překládal CIPE. Po nainstalování zdrojů a rozabalení jsem je zkompiloval spolu s ostatními moduly příkazem

```
# export PATCH_THE_KERNEL=NO
# make-kpkg clean
# make-kpkg --append-to-version -yoda --revision 1 modules_image
```

Nainstalovaný modul pěkně funguje.

Po nějaké době jsem potřeboval použít jádro `2.4.22`. CIPE ovšem nešel s tímto jádrem přeložit. Brzy jsem zjistil, že jádro `2.4.22` v Debianu není čisté *vanilla* jádro od Linuse, ale je vývojáři Debianu upraveno. Zcela jistě jsou do něj portovány nějaké věci z jader `2.6.0-pre`. Vyvstal problém jak CIPE s tímto jádrem přeložit. Možnost použití *vanilla* jádra jsem si ponechal až jako poslední možnost a jal se hledat řešení.

Upravil jsem soubor `/usr/src/modules/cipe/cipe/output.c` následujícím způsobem:

* ***$ diff -u output.c.orig output.c***

```
patch:cipe-1.5.4free-6_for_kernel-2.4.22_part2;
```

Zbývalo dořešit jakým způsobem a kde definuji identifikátor `LINUX_26` který oznamuje že se jedná o jádro do kterého byly přepsány některé věci z jádra `2.6.0-pre`. První pokus byl zavést tento identifikátor v souboru `/usr/src/linux/include/linux/version.h`.

```
#define LINUX_26 1
```

Tento soubor se ovšem generuje a tak opravdu není nejvhodnějším místem. Poté jsem se rozhodl pro „špinavý“ hack. Definici jsem přidal do souboru `/usr/src/modules/cipe/cipe/cipe.h`

```
patch:cipe-1.5.4free-6_for_kernel-2.4.22_part2;
```

Jak je vidět definici symbolu `LINUX_26` jsem podmínil verzí jádra `2.4.22`, neboť nevím jak zjistit že se jedná o upravené jádro pro Debian a počítám s tím že budu překládat jen toto upravené jádro.

27.1.2. Příprava Linux 2.4.24-3 + CIPE 1.5.4-free7

Poznámka: Použité verze programů jsou aktuální k 2004-02-26

Zdroje jádra jsou staženy z backports

```
deb http://www.backports.org/debian woody kernel-source-2.4.24 kernel-package
```

```
# apt-get source cipe-source
```

Po nainstalování zdrojů a nezbytných programů rozbalíme jádro.

```
# tar xjf kernel-source-2.4.24.tar.bz2
# ln -sf kernel-source-2.4.24 linux
```

```
# batch
warning: commands will be executed using /bin/sh
at> cd /usr/src/linux
at> export PATCH_THE_KERNEL=NO
at> make-kpkg clean
at> Ctrl-D
```

```
# export PATCH_THE_KERNEL=NO
# make-kpkg --append-to-version -tongu --revision 2 --config menu kernel_image modules_image
```

Nahrál jsem starší konfiguraci z `moon.1`, uložil do `moon.2` a ukončil konfiguraci s uložením. Tím se spustil překlad jádra.

27.1.3. Linux 2.4.27 + cipe 1.5.4-free7

Ukázka přípravy jádra 2.4.27 s cipe pro směrovač `brona` na stroji `deb`.

Debianí zdroje `/etc/apt/sources.list`:

```
### sources.list pro apt-proxy

# Local packages
deb file:/root/debs ./
```

```
# Apt-proxy cache is on host with domain cname debian running on port 9999
deb http://debian:9999/main woody main contrib
deb http://debian:9999/non-US woody/non-US main contrib
deb http://debian:9999/security stable/updates main

deb http://localhost:9999/backports woody kernel-package modutils
deb http://localhost:9999/backports woody kernel-source-2.4.26
deb http://localhost:9999/backports woody kernel-source-2.4.27

deb-src http://debian:9999/main sarge main

deb http://debian:9999/main sarge main

# apt-get update

deb:/usr/src# apt-get install cipe-source
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  cipe-source
0 packages upgraded, 1 newly installed, 0 to remove and 0  not upgraded.
Need to get 0B/188kB of archives. After unpacking 246kB will be used.
Reading changelogs...
Selecting previously deselected package cipe-source.
(Reading database ... 11497 files and directories currently installed.)
Unpacking cipe-source (from .../cipe-source_1.5.4free-9_all.deb) ...
Setting up cipe-source (1.5.4free-9) ...

deb:/usr/src# apt-get install kernel-source-2.4.27
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  kernel-source-2.4.27
0 packages upgraded, 1 newly installed, 0 to remove and 0  not upgraded.
Need to get 31.2MB of archives. After unpacking 31.3MB will be used.
Get:1 http://localhost woody/kernel-source-2.4.27 kernel-source-2.4.27 2.4.27-5 [31.2MB]
Fetched 31.2MB in 59s (521kB/s)
Reading changelogs...
Selecting previously deselected package kernel-source-2.4.27.
(Reading database ... 11503 files and directories currently installed.)
Unpacking kernel-source-2.4.27 (from .../kernel-source-2.4.27_2.4.27-5_all.deb) ...
Setting up kernel-source-2.4.27 (2.4.27-5) ...
```

Jádro sestavíme skriptem `/root/build-kernel-brona`

```
#!/bin/sh

#export PATCH_THE_KERNEL=YES
#export MAKEFLAGS="CC=gcc-3.0"

cd /usr/src
rm -fr modules

# Prepare CIPE source
tar xzf cipe.tar.gz

# Prepare kernel source
rm -fr kernel-source-2.4.27
```

```
tar xjf kernel-source-2.4.27.tar.bz2
rm linux
ln -s kernel-source-2.4.27 linux

# Copy config
cp /root/kernel/2.4.27/brona* linux

cd linux
make-kpkg clean
make-kpkg --append-to-version -brona --revision $1 \
          --config menu kernel_image modules_image
/root/build-kernel-brona
```

Skript spouštíme s parametrem udávajícím číslo verze

```
deb:/usr/src# nice -n19 ~/build-kernel-brona 1
```

Po úspěšném překladu nám vzniknou dva balíčky

```
deb:/usr/src# ls -l *-brona_*
-rw-r--r--    1 root     src         33316 Oct  4 14:49 cipe-2.4.27-brona_1.5.4free-9+1_i386.d
-rw-r--r--    1 root     src       2139164 Oct  4 14:49 kernel-image-2.4.27-brona_1_i386.deb
```

tyto balíčky přeneseme na cílový počítač a tam je nainstalujeme.

```
brona:~# stuff/update-debs
 ** Packages in archive but missing from override file: **
  cipe-2.4.27-brona kernel-image-2.4.27-brona

 Wrote 2 entries to output Packages file.

brona:~# apt-get install kernel-image-2.4.27-brona
```

Tím máme připravené jádro s modulem.

Jako další si připravíme balíček `cipe-common`:

```
deb:/usr/src/deb/cipe# apt-get source cipe-common
Reading Package Lists... Done
Building Dependency Tree... Done
Need to get 193kB of source archives.
Get:1 http://debian sarge/main cipe 1.5.4free-9 (dsc) [617B]
Get:2 http://debian sarge/main cipe 1.5.4free-9 (tar) [139kB]
Get:3 http://debian sarge/main cipe 1.5.4free-9 (diff) [53.9kB]
Fetched 193kB in 56s (3392B/s)
dpkg-source: extracting cipe in cipe-1.5.4free

Get:1 http://debian woody/main gettext-el 0.10.40-5 [41.9kB]
Get:2 http://debian woody/main gettext 0.10.40-5 [340kB]
Get:3 http://debian sarge/main intltool-debian 0.30+20040213 [23.5kB]
Get:4 http://debian sarge/main po-debconf 0.8.13 [65.1kB]
```

27.1.4. Konfigurace

FIXME: Popis jak konfigurovat

27.1.5. Ukázková konfigurace

Mějme následující konfiguraci tří směrovačů:

pikara

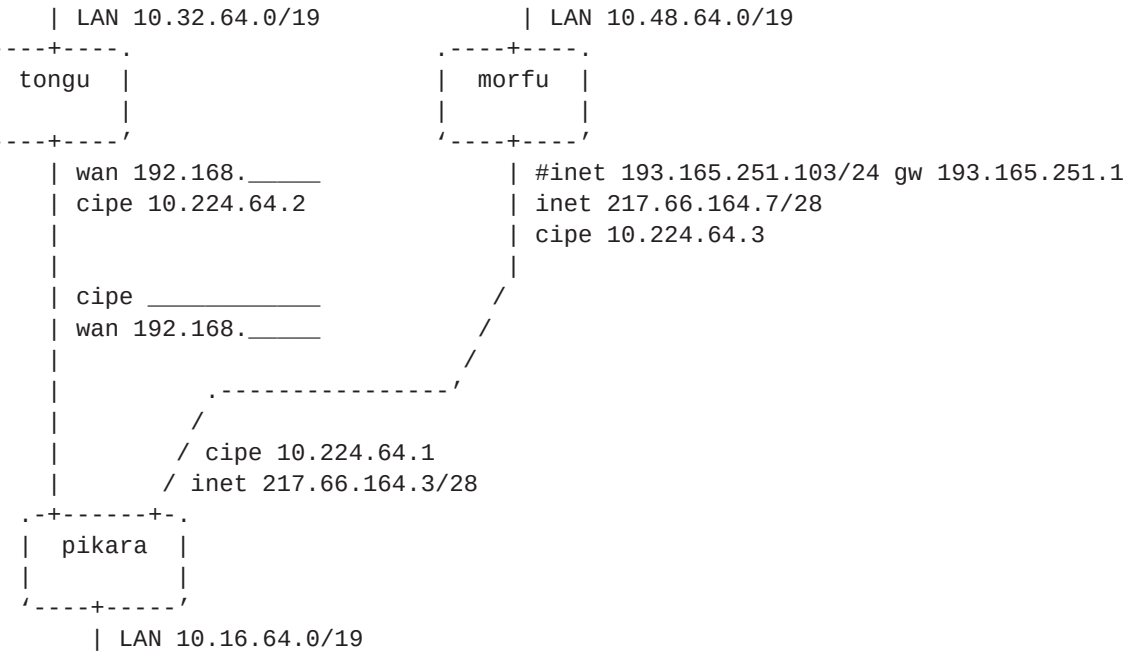
Centrální VPN směrovač v sídle firmy. Síť firmy 10.16.0.0/13

tongu

Vzdálený VPN směrovač na pobočce. Síť firmy `10.32.0.0/13`

morfu

Vzdálený VPN směrovač na druhé pobočce. Síť pobočky `10.48.64.0/19`



Konfigurace (úpravy v konfiguraci) jednotlivých strojů

27.1.5.1. pikara

Příklad 27-1. pikara:/etc/cipe/peers/morfu

```
# Link to morfu

arg     morfu

ipaddr  10.224.64.1
itpaddr 10.224.64.3

me      217.66.164.3:6789
peer    217.66.164.7:6789


maxerr  -1
key     Tajný klíč společný pro VPN se směrovačem morfu
```

Příklad 27-2. pikara:/etc/cipe/ip-up

```
case $6❶ in
        tongu)
                route add -net 10.32.0.0/13 gw $5
                ;;
        morfu)
                route add -net 10.48.0.0/13 gw $5
                ipchains -A input -p udp -s 217.66.164.7 6789 \
                        -d 217.66.164.3 6789 -i eth2 -j ACCEPT
                ;;
esac
```

❶ Jako parametr `$6` je předáno jméno skriptu, tedy jméno protějšího směrovače.

27.1.5.2. tongu

.

27.1.5.3. morfu

Příklad 27-3. morfu:/etc/cipe/peers/pikara

```
# Link to pikara

arg     pikara

ipaddr  10.224.64.3
ptpaddr 10.224.64.1

#me      193.165.251.103:6789
me      217.66.164.7:6789
peer    217.66.164.3:6789

maxerr  -1
       key     Tajný klíč společný pro VPN se směrovačem morfu❶
```

❶ Stejný tajný klíč jako je v 27-1

Příklad 27-4. `morfu:/etc/cipe/ip-up`

```
route add -net 10.0.0.0/8 gw $5
ipchains -A input -s 10.48.0.0/13 -i $1 -j ACCEPT
```

27.2. IPsec

Odkazy

- Kapitola IPSEC: secure IP over the Internet (http://lartc.org/howto/lartc.ipsec.html) z dokumentu Linux Advanced Routing & Trafic Control HOWTO (http://lartc.org/howto/index.html)
- IPsec-Tools (http://ipsec-tools.sourceforge.net/), IPsec Tools (http://sourceforge.net/projects/ipsec-tools) — User-space IPsec tools for various IPsec implementations. A port of KAME's libipsec, setkey, and racoon to the Linux OS. Also works on various BSD systems.
- Série článků Tuneluji, tuneluješ, tunelujeme. (http://www.root.cz/print.php4?id=1715) na root.cz (http://www.root.cz)
- . ()

FIXME:

IPsec byl implementován do Linuxového jádra od verze 2.5.40. Herbert Xu zahrnul „backport" této implementace do jader řady 2.4 v Debianu. Zcela jistě se vyskytuje v jádru 2.4.25 a novějších. Pro její zprovoznění je třeba mít přeložené jádro s volbami: `CONFIG_INET_AH`, `CONFIG_INET_ESP`, `CONFIG_XFRM` a `CONFIG_XFRM_USER`. Tyto části mohou být zakompilovány přimo do jádra, nebo mohou být přeloženy jako moduly.

27.2.1. Něco málo o tom jak to funguje

Odkazy

- Dokumentace k balíčku [xref linkend="deb-package.ipsec-tools"/]
- uvod do ipsec (protokoly, implementace, pouziti) (http://hysteria.sk/prielom/17/#5)
- IP Security Protocol (ipsec) (http://www.ietf.org/html.charters/ipsec-charter.html)
- rfc:2401;
- . ()

Základními stavebními kameny IPsec jsou dva protokoly:

- AH — *Authentication Header*
- ESP — *Encapsulated Security Payload*

První (AH) se stará o autentizaci, a zajišťuje že víme s kým komunikujeme. Druhý (ESP) pak provádí, mimo autentizaci, šifrování obsahu komunikace a zajišťuje že do naší komunikace nikdo nevidí.

Mimo tyto protokoly máme ještě:

- IKE — *Internet Key Exchange*

Slovník pojmů

ah

.

esp

FIXME:

IKE — *Internet Key Exchange*

FIXME:

isakmp

FIXME:

SA — *Security Association*

FIXME:

SAD — *Security Association Database*

FIXME:

SPD — *Security Policy Database*

FIXME:

FIXME:

27.2.1.1. Transportní režim

FIXME:

```
IPsec transport mode

        HOST-A ================ HOST-B
        (A)                    (B)

        IKE negotiation: A <--> B
        phase 1 ID payloads: <anything, anything>
        SA addresses: A <--> B
        outgoing packet: IP(A->B)
        phase 2 ID payloads: none, or <A, B>

        HOST-A's policy:
                spdadd A B any -P out ipsec ah/transport//require;
                spdadd B A any -P in ipsec ah/transport//require;

        HOST-B's policy:
                spdadd B A any -P out ipsec ah/transport//require;
                spdadd A B any -P in ipsec ah/transport//require;

        both racoon.conf:
                no particular twists
```

27.2.1.2. Tunelovací režim

FIXME:

```
IPsec tunnel mode

        HOST-A --- Gateway-A =========== Gateway-B --- HOST-B
        (A)        (GA)                 (GB)          (B)

        IKE negotiation: GA <--> GB
        phase 2 ID payloads: <anything, anything>
                IDs should reflect GA and GB's authenticity.
        SA addresses: GA <--> GB
        outgoing packet: IP(GA->GB)
```

```
phase 2 ID payloads: A, B

Gateway-A's policy:
        spdadd A B any -P out ipsec esp/tunnel/GA-GB/require;
        spdadd B A any -P in ipsec esp/tunnel/GB-GA/require;

Gateway-B's policy:
        spdadd B A any -P out ipsec esp/tunnel/GB-GA/require;
        spdadd A B any -P in ipsec esp/tunnel/GA-GB/require;

both racoon.conf:
        no particular twists
```

27.2.2. Ukázka instalace a konfigurace IPsec

Odkazy

- Debian Backports (Backports.ORG) (http://www.backports.org/)
- . ()
- . ()

27.2.2.1. Příprava jádra s podporou IPsec

Přípravu provádím na virtuálním stroji `bone` na kterém je čistá instalace Debian/GNU Linux Woody. Protože je stroj holý, musím nainstlaovat vše potřebné. Budu potřebovat i balíčky z Debian Backports (Backports.ORG) (http://www.backports.org/), upravím si tedy zdroje v `/etc/apt/sources.list` takto:

```
deb http://localhost:9999/main woody main contrib non-free
deb http://localhost:9999/non-US woody/non-US main contrib non-free
deb http://localhost:9999/security woody/updates main non-free

deb http://localhost:9999/backports woody kernel-package modutils
deb http://localhost:9999/backports woody kernel-source-2.4.27
```

```
# apt-get update
# apt-get upgrade
⋮
The following packages will be upgraded
  gzip libsasl7 login passwd wget
⋮
```

Poté nainstaluju postupně všechny potřebné balíčky.

Instaluji: `kernel-source-2.4.27` (2.4.27-5), `dpkg-dev` (1.9.21), `kernel-package` (7.107), `libncurses5-dev` (5.2.20020112a-7), `gcc`

Závislosti: `binutils` (2.12.90.0.1-4), `make` (3.79.1-14), `patch` (2.5.4-11), `perl` (5.6.1-8.7), `perl-modules` (5.6.1-8.7), `libc6-dev` (2.2.5-11.5), `cpp` (2.95.4-14), `cpp-2.95` (2.95.4-11woody1), `gcc` (2.95.4-14), `gcc-2.95` (2.95.4-11woody1)

* *Budu možná potřebovat balíčky: debhelper*

Po nainstalování ještě dokonfiguruju některé balíčky. U `kernel-package` jsem upravil soubor `/etc/kernel-pkg.conf`

```
# The maintainer information.
maintainer := Radek Hnilica
email := radek@hnilica.cz
```

Poté pokračuji vlastní konfigurací jádra

```
# cd /usr/src
# tar xjf kernel-source-2.4.27.tar.bz2
# ln -s kernel-source-2.4.27 linux
# cd /usr/src/linux
# make menuconfig
```

První krok je nahrání konfigurace z některého ze stávajících routerů. Vybral jsem si router `tongu` na kterém je jádro 2.4.24. Po nahrání konfigurace nastavím:

`CONFIG_INET_AH`
Networking options / IP: AH transformation

Support for IPsec ESP.

Nastavím na Y

`CONFIG_INET_ESP`
Networking options / IP: ESP transformation

Support for IPsec ESP.

Nastavím na Y

`CONFIG_XFRM_USER`
Networking options / IP: IPsec user configuration interface

Support for IPsec user configuration interface used by native Linux tools..

Nastavím na Y

`CONFIG_XFRM`
Networking options / IP: IPsec user configuration interface

Support for IPsec user configuration interface used by native Linux tools.

Nastavit na `Y`.

`CONFIG_INET_IPCOMP`

Support for IP Paylod Compression (RFC3173), typically needed for IPsec.

Nastavil jsem na `Y`.

Poté jádro přeložíme

```
bone:/usr/src/linux# make-kpkg clean
# make-kpkg --append-to-version -router --revision $(date +%Y%m%d.%H%M) \
            kernel_image modules_image
```

* *Na Debian Backports (Backports.ORG) (http://www.backports.org/) jsou ke dnešnímu dni (2004-12-01) zdroje jader 2.4.25, 2.4.26, 2.4.27, 2.6.6 a 2.6.7*

27.2.2.2. Příprava nástrojů

Nejdříve přidáme do souboru `/etc/apt/sources.list`

```
deb http://localhost:9999/backports woody po-debconf automake1.7 autoconf
deb-src http://localhost:9999/backports woody ipsec-tools
```

Nainstalujeme všechny potřebné balíčky

```
# apt-get install flex bison libssl-dev debhelper libtool autoconf po-debconf automake1.7
# apt-get install kernel-headers-2.4.18
```

*

A provedempe překlad

```
bone:/usr/src/debs# apt-get update
bone:/usr/src/debs# apt-get source --build ipsec-tools
```

27.2.2.3. Instalace routeru

FIXME:

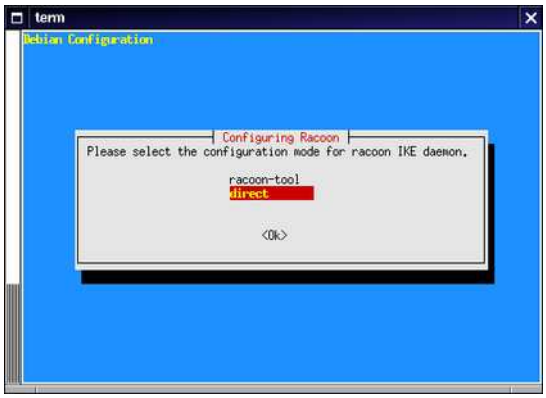
```
router# apt-get install ipsec-tools racoon
```

Po jádře si musíme připravit nástroje. Ty jsou k dispozici na Debian Backports (Backports.ORG) (http://www.backports.org/) a proto si jen přidáme zdroj do `/etc/apt/sources.list`

```
deb http://debian:9999/backports debian ipsec-tools
```

```
pikachu:~# apt-get update
pikachu:~# apt-get install ipsec-tools racoon
```

Nastavím `racoon` při instalaci:



Bezpečnostní asociace (*security association*) pro AH

```
add 10.0.0.11 10.0.0.216 ah 15700 -A hmac-md5 "1234567890123456";
```

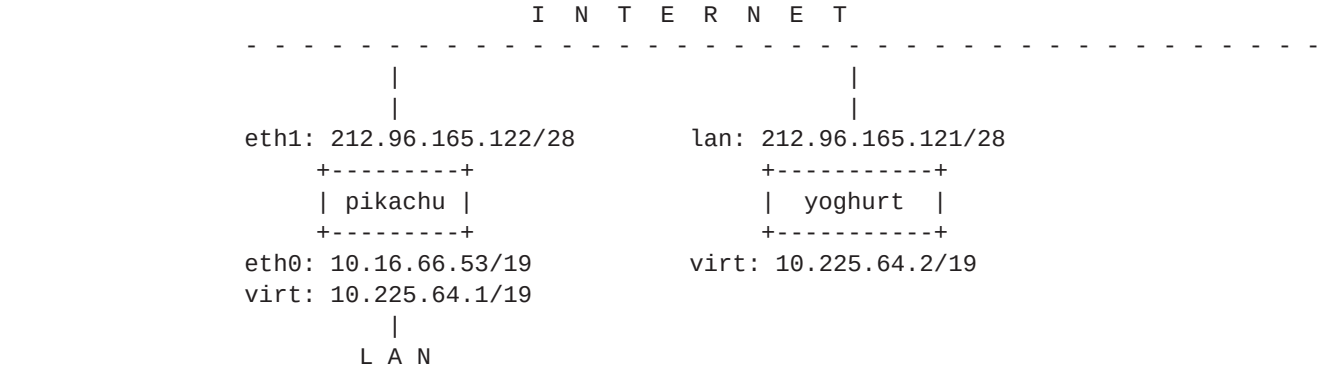
SA pro ESP

```
add 10.0.0.11 10.0.0.216 esp 15701 -E 3des-cbc "123456789012345678901234";
```

```
# modprobe af_key
```

27.2.2.4. Propojení dvou linuxů v transportním režimu s užitím certifikátů

Obrázek 27-1. Konfigurace Linux Host — Linux Host



Připravil jsem si všechny certifikáty. Jak certifikační autoritu (`ca.crt`), tak pro oba stroje (`pikachu` a `yoghurt`)

```
$ openssl req -new nodes -newkey rsa:1024 -sha1 -keyout ca.key -out ca.req
$ openssl x509 -req -days 9999 -in ca.req -signkey ca.key -out ca.crt
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyout pikachu.key \
          -out pikachu.req
$ openssl x509 -req -days 370 -in pikachu.req -out pikachu.crt \
          -CA ca.crt -CAkey ca.key -CAcreateserial
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyout yoghurt.key \
          -out yoghurt.req
$ openssl x509 -req -days 370 -in yoghurt.req -out yoghurt.crt \
          -CA ca.crt -CAkey ca.key -CAcreateserial
```

Na straně hosta `pikachu` je následující konfigurace. Soubor `/etc/racoon/racoon.conf`:

```
# Konfigurace s pouzitim certifikatu

#log debug;     #notify/debug/debug2
log notify;

path certificate "/etc/racoon/certs";
path pre_shared_key "/etc/racoon/psk.txt";

listen {
        isakmp 212.96.165.122;
}

remote anonymous
```

```
{
        exchange_mode aggressive,main;
        doi ipsec_doi;
        situation identity_only;

        lifetime time 24 hour;          # min/hour
        initial_contact on;

        my_identifier asn1dn;
        peers_identifier asn1dn;
        certificate_type x509 "pikachu.crt" "pikachu.key";

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                #authentication_method rsasig;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo anonymous {
        pfs_group modp1024;
        lifetime time 1 hour;           # min/hour
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

V souboru `/etc/racoon/spd.conf` je

```
#!/usr/sbin/setkey -f
# pikachu:/etc/racoon/ipsec.conf
# Spojeni s yoghurt,jirkanb,trada

### Flush all
flush;
spdflush;

### Security Policy
# TUNNEL MODE
spdadd 10.16.64.0/19 10.225.64.0/19 any -P out ipsec
        esp/tunnel/10.16.66.53-10.225.64.3/require;
spdadd 10.225.64.0/19 10.16.64.0/19 any -P in ipsec
        esp/tunnel/10.225.64.3-10.16.66.53/require;

# Radek Yoghurt
spdadd 212.96.165.122 212.96.165.121 any -P out ipsec
        esp/transport//require;
spdadd 212.96.165.121 212.96.165.122 any -P in ipsec
        esp/transport//require;


# trada.firma.cz from gprs: 160.218.179.137
spdadd 212.96.165.122 160.218.179.137 any -P out ipsec
        esp/transport//require;
spdadd 160.218.179.137 212.96.165.122 any -P in ipsec
```

```
        esp/transport//require;

# trada.firma.cz from lan: 212.96.165.120/28
spdadd 212.96.165.122 212.96.165.120 any -P out ipsec
        esp/transport//require;
spdadd 212.96.165.120 212.96.165.122 any -P in ipsec
        esp/transport//require;
```

A certifikáty

```
pikachu:/etc/racoon# ls -l /etc/racoon/certs/
total 12
lrwxr-xr-x    1 root     root            6 Jan  3 13:07 6686505c.0 -> ca.crt
-rw-r--r--    1 root     root          985 Jan  3 13:00 ca.crt
-rw-r--r--    1 root     root          973 Jan  3 13:02 pikachu.crt
-rw-r--r--    1 root     root          887 Jan  3 13:02 pikachu.key
pikachu:/etc/racoon#
```

Na straně hosta `yoghurt` je konfigurace následující. V souboru `/etc/racoon/racoon.conf` je

```
# Konfigurace s pouzitim certifikatu

log debug2;

path  certificate "/etc/racoon/certs";
path pre_shared_key "/etc/racoon/psk.txt";

remote anonymous
{
        exchange_mode aggressive,main;
        doi ipsec_doi;
        situation identity_only;

        lifetime time 2 min;
        initial_contact on;
        proposal_check obey;

        my_identifier asn1dn;
        peers_identifier asn1dn;
        certificate_type x509 "yoghurt.crt" "yoghurt.key";

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo anonymous
{
        pfs_group modp1024;
        lifetime time 5 min;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

V souboru `/etc/racoon/spd.conf` je:

```
#!/usr/sbin/setkey -f

flush;
spdflush;

spdadd 212.96.165.121 212.96.165.122 any -P out ipsec
        esp/transport//require;
spdadd 212.96.165.122 212.96.165.121 any -P in ipsec
        esp/transport//require;
```

A certifikáty:

```
yoda:/volume/d1/etc/racoon# ls -l certs/
total 12
lrwxr-xr-x    1 root     root            6 Jan  3 14:09 6686505c.0 -> ca.crt
-rw-r--r--    1 root     root          985 Jan  3 13:04 ca.crt
-rw-r--r--    1 root     root          956 Jan  3 13:04 yoghurt.crt
-rw-r--r--    1 root     root          891 Jan  3 13:04 yoghurt.key
yoda:/volume/d1/etc/racoon#
```

Symbolický odkaz na certifikát `ca.crt` vytvoříme příkazem:

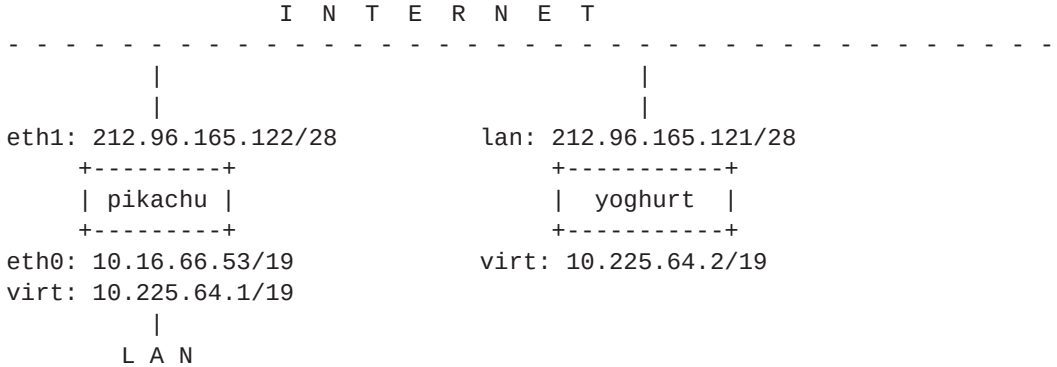
```
# ln -s ca.crt $(openssl x509 -noout -hash -in ca.crt).0
```

Spojení funguje. Z jednoho stroje pingám na druhý.

27.2.2.5. Propojení dvou linuxů v tunelovacím režimu s užitím certifikátů

Vycházím s předchozí konfigurace. Tedy použiji stejné certifikáty.

Obrázek 27-2. Konfigurace Linux Host — Linux Host



Na straně hosta `pikachu` je následující konfigurace. Soubor `/etc/racoon/racoon.conf`:

```
# Konfigurace s pouzitim certifikatu

#log debug;     #notify/debug/debug2
log notify;

path certificate "/etc/racoon/certs";
path pre_shared_key "/etc/racoon/psk.txt";
```

```
listen {
        isakmp 212.96.165.122;
}

remote anonymous
{
        exchange_mode aggressive,main;
        doi ipsec_doi;
        situation identity_only;

        lifetime time 24 hour;          # min/hour
        initial_contact on;

        my_identifier asn1dn;
        peers_identifier asn1dn;
        certificate_type x509 "pikachu.crt" "pikachu.key";

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                #authentication_method rsasig;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo anonymous {
        pfs_group modp1024;
        lifetime time 1 hour;           # min/hour
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

V souboru `/etc/racoon/spd.conf` je

```
#!/usr/sbin/setkey -f
# pikachu:/etc/racoon/ipsec.conf

### Flush all
flush;
spdflush;

### Security Policy
spdadd 212.96.165.122 212.96.165.121 any -P out ipsec
        esp/transport//require;
spdadd 212.96.165.121 212.96.165.122 any -P in ipsec
        esp/transport//require;
# TUNNEL MODE
spdadd 10.16.64.0/19 10.225.64.0/19 any -P out ipsec
        esp/tunnel/212.96.165.122-212.96.165.121/require;
spdadd 10.225.64.0/19 10.16.64.0/19 any -P in ipsec
        esp/tunnel/212.96.165.121-212.96.165.122/require;
```

Na straně hosta `yoghurt` je konfigurace následující. V souboru `/etc/racoon/racoon.conf` je

```
# Konfigurace s pouzitim certifikatu
```

```
log debug2;

path  certificate "/etc/racoon/certs";
path pre_shared_key "/etc/racoon/psk.txt";

remote anonymous
{
        exchange_mode aggressive,main;
        doi ipsec_doi;
        situation identity_only;

        lifetime time 2 min;
        initial_contact on;
        proposal_check obey;

        my_identifier asn1dn;
        peers_identifier asn1dn;
        certificate_type x509 "yoghurt.crt" "yoghurt.key";

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo anonymous
{
        pfs_group modp1024;
        lifetime time 5 min;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

V souboru `/etc/racoon/spd.conf` je:

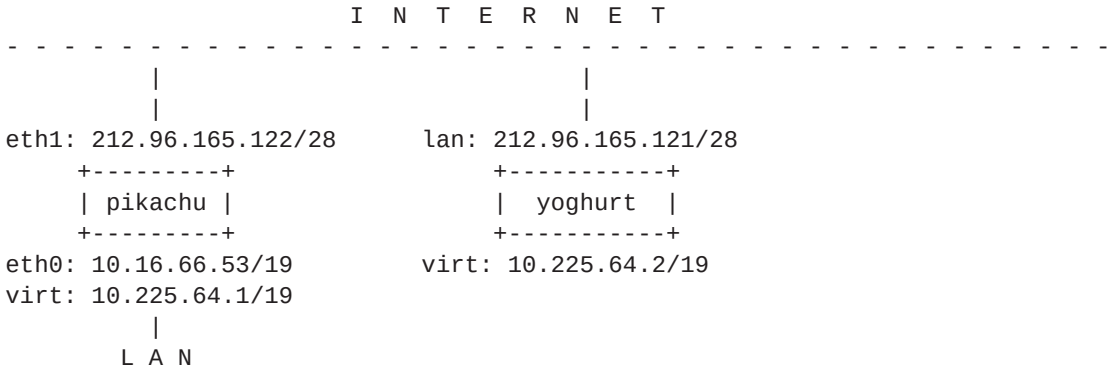
```
#!/usr/sbin/setkey -f

flush;
spdflush;

spdadd 212.96.165.121 212.96.165.122 any -P out ipsec
        esp/transport//require;
spdadd 212.96.165.122 212.96.165.121 any -P in ipsec
        esp/transport//require;

spdadd 10.225.64.0/19 10.16.64.0/19 any -P out ipsec
        esp/tunnel/212.96.165.121-212.96.165.122/require;
spdadd 10.16.64.0/19 10.225.64.0/19 any -P out ipsec
        esp/tunnel/212.96.165.122-212.96.165.121/require;
```

Spojení nefunguje.

27.2.2.6. Konfigurace linux road warior - linux access router

Ukázková konfigurace

Obrázek 27-3. Konfigurace Linux RoadWarior — Linux AccessRouter



Následující poznámky jsou z IPsec HOWTO (http://autistici.org/ginox/ipsec-howto/ipsec-howto.html) část Road Warrior.

Na straně road warriora bude konfigurace v souboru `/etc/racoon/racoon.conf` následující:

```
path certificate "/etc/certs";

remote anonymous {
        exchange_mode main;
        generate_policy on;
        passive on;
        certificate_type x509 "my_certificate.pem" "my_private_key.pem";
        my_identifier asn1dn;
        peers_identifier asn1dn;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method rsasig;
                dh_group modp1024;
        }
}


sainfo anonymous {
        pfs_group modp1024;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Příkazy zvírazněné v předchozí konfiguraci jsou důležité.

```
generate_policy on;
```

Ze stránky IPsec Road Warrior Configuration (http://ou800doc.caldera.com/en/NET_ipsec/roadwarrior.html) pak vybírám:

Na straně serveru nastavíme v souboru `/etc/inet/ipsec.conf`:

```
pdadd server-ip-address 0.0.0.0 any -P out ipsec
```

```
        esp/tunnel/server-ip-address-0.0.0.0/require;
spdadd 0.0.0.0 server-ip-address any -P in ipsec
        esp/tunnel/0.0.0.0-server-ip-address/require;
```

V souboru `/etc/inet/racoon.conf` je pak:

```
path pre_shared_key "/etc/inet/psk.txt" ;
#
timer {
 phase1 60 seconds ;
 phase2 60 seconds ;
}
#
remote anonymous {
        exchange_mode main, aggressive, base ;
 doi ipsec_doi ;
 situation identity_only ;
 lifetime time 1 hour ;
        generate_policy on;
        passive on;
        my_identifier address server-ip-address ;
 peers_identifier fqdn "domain-name" ;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
 proposal_check obey ;
}
#
sainfo anonymous {
        pfs_group modp1024;
 lifetime time 1 hour ;
 encryption_algorithm 3des, blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
        compression_algorithm deflate;
}
```

Článek (http://ou800doc.caldera.com/en/NET_ipsec/roadwarrior.html) dále popisuje konfiguraci programu *SonicWALL VPN Client* na MSWindows jako road warriora. Vypadá že popisuje moji situaci.

27.2.3. Poznámky k `racoon`

Konfigurační a jiné soubory:

`psk.txt`

Soubor obsahuje sdílené klíče používané metodou *Pre-shared key authentication*. Soubor obsahuje páry (identifikátor, sdílený klíč). Každý pár je na samostatném řádku. V prvním sloupci je identifikátor a ten je libovolným počtem mezer či tabelátorů oddělen od druhého sloupce, kde je klíč. Řádky začínající znakem '#' jsou považovány za komentář a ignorovány. Klíč který začíná znaky '`0x`' je interpretován jako hexadecimální číslo.

Poznámka: Vlastníkem souboru musí být proces racoon a tento soubor musí být čitelný jen pro něj. T.j. musí být chránen před ostatními uživateli systému.

```
pikachu:/etc/racoon# chmod 400 psk.txt
pikachu:/etc/racoon# ls -l psk.txt
-rw-------    1 root     root          140 Dec  6 11:07 psk.txt
```

Ukázka obsahu souboru převzatá z manuálových stránek:

```
10.160.94.3     mekmitasdigoat
172.16.1.133    0x12345678
194.100.55.1    whatcertificatereally
3ffe:501:410:ffff:200:86ff:fe05:80fa    mekmitasdigoat
3ffe:501:410:ffff:210:4bff:fea2:8baa    mekmitasdigoat
foo@kame.net    mekmitasdigoat
foo.kame.net    hoge
```

`/etc/racoon/racoon.conf`

Konfigurační soubor pro démona `racoon`

Ukázkový soubor podle IPsec HOWTO (http://www.ipsec-howto.org/x247.html)

```
path pre_shared_key "/etc/psk.txt";

remote 192.168.2.100 {
        exchange_mode main;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo address 172.16.1.0/24 any address 172.16.2.0/24 any {
        pfs_group modp768;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

`/etc/racoon/racoon-tool.conf`

–

FIXME:

```
# How to control the syslog level
global:
        log: notify

# %default configuration to avoid duplication
peer(%default):
        verify_identifier: on
        hash_algorithm[0]: sha1
        encryption_algorithm[0]: aes
connection(%default):
        src_ip: 212.96.165.122
        src_range:      10.16.64.0/19
        dst_range:      10.225.96.0/19
```

```
## Road Warior Jirka
peer(160.218.179.137):
        peers_identifier: address
connection(jirka):
        dst_ip: 160.218.179.137
        admin_status: enabled
```

Šifrovací algoritmy: aes

Vygenerování klíče:

```
$ dd if=/dev/random count=20 bs=1 | xxd -ps
```

Program **xxd** je součástí balíčku `vim`.

Různě posbírané informace k některým parametrům v konfiguračním souboru `/etc/racoon/racoon-tool.conf`:

`sainfo`

sainfo adress . . .

sainfo adress xxx.xxx.37.118[any] any address 192.168.1.0/24[any] any
. . .

sainfo adress 192.168.1.0/24[any] any address xxx.xxx.37.118[any] any
. . .

Význam parametru neznám.

Specifikace sainfo definuje parametry druhé fáze IKE. (IPsec-SA establishment). Následující popis je z `racoon.conf(5)`:

`sainfo (`*`source_id destination_id`*` | anonymous) [from `*`idtype`*` [`*`string`*`]] {`*`statements`*`}`

encryption_algorithm

–

Algoritmy: des, 3des, des_iv64, des_iv32, rc5, rc4, idea, 3idea, cast128, blowfish, null_enc, twofish, rijndael (used with ESP)

authentication_algorithm

–

Algoritmy: des, 3des, des_iv64, des_iv32, hmac_md5, hmac_sha1, non_auth (used with ESP)

compression_algorithm

–

Algoritmy: deflate (used with IPComp)

.

–

Logging level.

log *level*;

Definuje úroveň logování. Přípustné hodnoty jsou: `notify`, `debug`, `debug2`. Implicitní hodnota je `notify`.

Poznámka: Na pomalejších strojích může vysoká úroveň logování způsobit neúspěch IKE potvrzování. Je to způsobeno překročením časových limitů.

27.2.3.1. Konfigurační soubor `racoon.conf`

FIXME:

certificate_type

Přípustné formy jsou:

```
certificate_type x509 certfile privkeyfile;
```

27.2.4. Pokus o spojení Linux - WinXP

Zkouším zprovoznit připojení Windows XP klienta k Linux routeru.

```
Dec  6 13:28:17 pikachu racoon: INFO: respond new phase 1 negotiation: 212.96.165.122[500]<=>160.218.179.137[500]
Dec  6 13:28:17 pikachu racoon: INFO: begin Identity Protection mode.
Dec  6 13:28:17 pikachu racoon: INFO: received Vendor ID: MS NT5 ISAKMPOAKLEY
Dec  6 13:29:18 pikachu racoon: ERROR: phase1 negotiation failed due to time up. e36427a42c8f6bf0:8cac0443e9f20d61
Dec  6 13:29:22 pikachu racoon: ERROR: unknown Informational exchange received.
Dec  6 13:29:57 pikachu racoon: INFO: respond new phase 1 negotiation: 212.96.165.122[500]<=>160.218.179.137[500]
Dec  6 13:29:57 pikachu racoon: INFO: begin Identity Protection mode.
Dec  6 13:29:57 pikachu racoon: INFO: received Vendor ID: MS NT5 ISAKMPOAKLEY
Dec  6 13:29:57 pikachu racoon: NOTIFY: the packet is retransmitted by 160.218.179.137[500].
Dec  6 13:29:58 pikachu racoon: NOTIFY: the packet is retransmitted by 160.218.179.137[500].
Dec  6 13:30:58 pikachu racoon: ERROR: phase1 negotiation failed due to time up. b7d6f2813dc2039c:7279a89d152d7d25
Dec  6 13:31:02 pikachu racoon: ERROR: unknown Informational exchange received.
.
.
.
Dec  6 13:32:55 pikachu racoon: INFO: respond new phase 1 negotiation: 212.96.165.122[500]<=>160.218.179.137[500]
Dec  6 13:32:55 pikachu racoon: INFO: begin Identity Protection mode.
Dec  6 13:32:55 pikachu racoon: INFO: received Vendor ID: MS NT5 ISAKMPOAKLEY
Dec  6 13:32:55 pikachu racoon: NOTIFY: the packet is retransmitted by 160.218.179.137[500].
Dec  6 13:32:56 pikachu racoon: NOTIFY: the packet is retransmitted by 160.218.179.137[500].
Dec  6 13:33:56 pikachu racoon: ERROR: phase1 negotiation failed due to time up. 058fd7944b79c793:4d8077a626f0ab4d
Dec  6 13:34:00 pikachu racoon: ERROR: unknown Informational exchange received.
```

Po výměně klíče, původně jsem pooužíval jako klíč hexadecimální číslo (začíná `0x`), za obyčejný textový řetězec, jsme pokročili. Tentokrát je chybové hlášení jiné:

```
Dec  6 13:52:50 pikachu racoon: INFO: respond new phase 1 negotiation: 212.96.165.122[500]<=>160.218.179.137[500]
Dec  6 13:52:50 pikachu racoon: INFO: begin Identity Protection mode.
Dec  6 13:52:50 pikachu racoon: INFO: received Vendor ID: MS NT5 ISAKMPOAKLEY
Dec  6 13:52:50 pikachu racoon: NOTIFY: the packet is retransmitted by 160.218.179.137[500].
Dec  6 13:52:51 pikachu racoon: NOTIFY: the packet is retransmitted by 160.218.179.137[500].
Dec  6 13:52:52 pikachu racoon: INFO: ISAKMP-SA established 212.96.165.122[500]-160.218.179.137[500] spi:00ce18b4eb2673c1:f947cc5701907350
Dec  6 13:52:54 pikachu racoon: INFO: respond new phase 2 negotiation: 212.96.165.122[0]<=>160.218.179.137[0]
Dec  6 13:52:54 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:52:54 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:52:54 pikachu racoon: ERROR: failed to pre-process packet.
Dec  6 13:52:55 pikachu racoon: INFO: respond new phase 2 negotiation: 212.96.165.122[0]<=>160.218.179.137[0]
Dec  6 13:52:55 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:52:55 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:52:55 pikachu racoon: ERROR: failed to pre-process packet.
```

```
Dec  6 13:52:58 pikachu racoon: INFO: respond new phase 2 negotiation: 212.96.165.122[0]<=>160.218.179.137[0]
Dec  6 13:52:58 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:52:58 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:52:58 pikachu racoon: ERROR: failed to pre-process packet.
Dec  6 13:53:01 pikachu /USR/SBIN/CRON[394]: (mail) CMD (  if [ -x /usr/sbin/exim -a -f /etc/exim/exim.conf ]; then /usr/sbin/exim -q ; fi)
Dec  6 13:53:03 pikachu racoon: INFO: respond new phase 2 negotiation: 212.96.165.122[0]<=>160.218.179.137[0]
Dec  6 13:53:03 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:53:03 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:53:03 pikachu racoon: ERROR: failed to pre-process packet.
Dec  6 13:53:11 pikachu racoon: INFO: respond new phase 2 negotiation: 212.96.165.122[0]<=>160.218.179.137[0]
Dec  6 13:53:11 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:53:11 pikachu racoon: ERROR: failed to get sainfo.
Dec  6 13:53:11 pikachu racoon: ERROR: failed to pre-process packet.
```

Mám podezření na sainfo. Že není dobře nakonfigurované, či vůbec nakonfigurované. Zvýšil jsem úroveň logování z `notify` na `debug`.

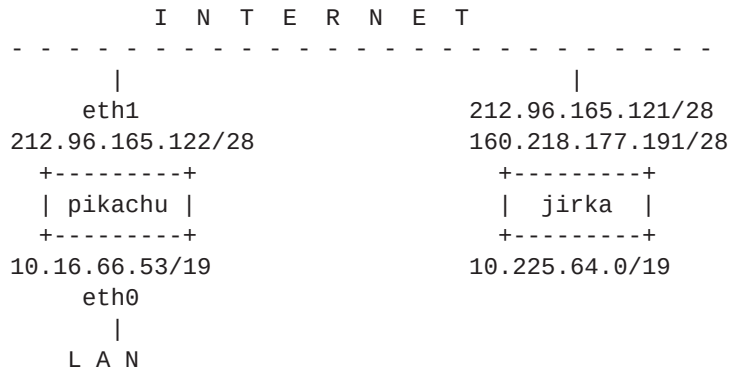
27.2.5. Experimentální konfigurace

* *$Header: /home/radek/cvs/unix-book/input/unix/sec-ipsec.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $*

Zde popisuji kompletní experiment, vytvoření IPsec spojení mezi serverem a jedním klientem.

Situace je následující. Mám server s vnitřní adresou `10.16.66.53/19` a vnější adresou `212.96.165.122`. Klient používá pevnou adresu `160.218.177.191`

Obrázek 27-4. Experimentální konfigurace pro IPsec



27.2.5.1. Vytvoření certifikátů

* *section condition="author"*
* ***FIXME:** zrušit!!!*

Pro vytvoření certifikátů potřebujeme mít nainstalováno `openssl`

```
pikachu:~# apt-get install openssl
```

FIXME:

```
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyform PEM -keyout \
          jirka.private -outform PEM -out request.pem

Country Name (2 letter code) [CZ]:CZ
State or Province Name (full name) []:.
Locality Name (eg, city) []:Breclav
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FIRMA a.s.
Organizational Unit Name (eg, section) []:laptop
```

```
Common Name (eg, YOUR name) []:Jiri Jiricek
Email Address []:jiri.jiricek@firma.cz
A challenge password []:
An optional company name []:
```

Vytvořený certifikát podepíšeme:

```
$ openssl x509 -req -in request.pem -signkey jirka.private \
      -out jirka.public
Signature ok
subject=/C=CZ/L=Breclav/O=FIRMA a.s./OU=laptop/CN=Jiri Jiricek/Email=jiri.jiricek@firma.cz
Getting Private key
```

Vytvoříme klíče pro server:

```
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyform PEM -keyout \
          pikachu.private -outform PEM -out request.pem

Country Name (2 letter code) [CZ]:CZ
State or Province Name (full name) []:.
Locality Name (eg, city) []:Breclav
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FIRMA a.s.
Organizational Unit Name (eg, section) []:IT Departement
Common Name (eg, YOUR name) []:Radek Hnilica
Email Address []:radek.hnilica@firma.cz
A challenge password []:
An optional company name []:
```

Vytvořený certifikát podepíšeme:

```
$ openssl x509 -req -in request.pem -signkey pikachu.private \
      -out pikachu.public
Signature ok
subject=/C=CZ/L=Breclav/O=FIRMA a.s./OU=IT Departement/CN=Radek Hnilica/Email=radek.hnilica@fi
Getting Private key
```

Tkto vytvořené certifikáty neměly úspěch. Na straně Windows klienta nešel naimportovat jeden z certifikátů.

27.2.5.1.1. Vytvoření certifikační autority

Vytvoření certifikátů podle Nate Carlson (http://www.natecarlson.com/linux/ipsec-x509.php). Nejdříve si připravíme certifikační autoritu. Na stroji který buse sloužit jako certifikační autorita nainstalujeme `openssl`. Pro experiment použiji přímo VPN server. V ostrém použití toto není vhodné a je lepší použít jiný stroj.

```
# apt-get install openssl
```

Nyní se v editoru otevřeme soubor `/etc/ssl/openssl.cnf` a opravíme konfiguraci. Opravil jsem standardní dobu platnosti certifikátů na 30 dní `default_days = 30` a hodnotu CRL na 5 dní `default_crl_days = 5`. V ostrém provozu je lépe použít více, např jeden až dva roky.

```
pikachu:~# /usr/lib/ssl/misc/CA.sh -newca
CA certificate filename (or enter to create)Enter

Making CA certificate ...
Using configuration from /usr/lib/ssl/openssl.cnf
Generating a 1024 bit RSA private key
```

```
.................................++++++
.......++++++
writing new private key to ’./demoCA/private/./cakey.pem’
Enter PEM pass phrase:certifikuj
Country Name (2 letter code) [CZ]:
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) [Breclav]:
Organization Name (eg, company) [FIRMA a.s.]:
Organizational Unit Name (eg, section) []:IT Departement
Common Name (eg, YOUR name) []:CA
Email Address []:ca@firma.cz
pikachu:~#
```

Nyní vygenerujeme `crl` soubor.

```
pikachu:~# openssl ca -gencrl -out crl.pem
Using configuration from /usr/lib/ssl/openssl.cnf
Enter PEM pass phrase:certifikuj
pikachu:~#
```

27.2.5.1.2. Vytvoření certifikátu

FIXME:

```
pikachu:~# /usr/lib/ssl/misc/CA.sh -newreq
Using configuration from /usr/lib/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.++++++
.................................................++++++
writing new private key to ’newreq.pem’
Enter PEM pass phrase:jirka.heslo
⋮
Country Name (2 letter code) [CZ]:
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) [Breclav]:
Organization Name (eg, company) [FIRMA a.s.]:
Organizational Unit Name (eg, section) []:IT Departement
Common Name (eg, YOUR name) []:Jiri Jiricek
Email Address []:jiri.jiricek@firma.cz
⋮
A challenge password []:Enter
An optional company name []:Enter
Request (and private key) is in newreq.pem
pikachu:~#
```

Vytvořený certifikát, který bychom si normálně nechali podepsat důvěryhodnou certifikační autoritou, např. Thawte nebo Verisign, si podepíšeme vlastní certifikační autoritou kterou jsem si vytvořili:

```
pikachu:~# /usr/lib/ssl/misc/CA.sh -sign
Using configuration from /usr/lib/ssl/openssl.cnf
Enter PEM pass phrase:certifikuj (heslo k certifikační autoritě)
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
```

```
countryName           :PRINTABLE:'CZ'
localityName          :PRINTABLE:'Breclav'
organizationName      :PRINTABLE:'FIRMA a.s.'
organizationalUnitName:PRINTABLE:'IT Departement'
commonName            :PRINTABLE:'Jiri Jiricek'
emailAddress          :IA5STRING:'jiri.jiricek@firma.cz'
Certificate is to be certified until Jan  7 09:31:34 2005 GMT (30 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
⋮
Signed certificate is in newcert.pem
pikachu:~#
```

Pro lepší orientaci si certifikáty přejmenujeme popisnými jmény.

```
pikachu:~# mv newcert.pem jiri.jiricek.pem
pikachu:~# mv newreq.pem jiri.jiricek.key
```

Pro potřeby počítače s OS MSWindows je třeba převést certifikáty do formátu p12

```
# openssl pkcs12 -export -in jiri.jiricek.pem -inkey jiri.jiricek.key  -certfile demoCA/cacert.pem -out
Enter PEM pass phrase:jirka.heslo (heslo k certifikátu)
Enter Export Password:p12heslo (heslo k p12 souboru)
Verifying password - Enter Export Password:p12heslo
pikachu:~#
```

Kvůli problémům s certifikáty jsem si je vytvořil znovu podle WAVElan Client setup for KAME/BSD (NetBSD, FreeBSD) (http://www.wavesec.org/kame.phtml).

```
mpressca@yoda:~$ openssl genrsa -out pikachu2.pem 1024
warning, not much extra random data, consider using the -rand option
Generating RSA private key, 1024 bit long modulus
.....++++++
.++++++
e is 65537 (0x10001)
mpressca@yoda:~$ openssl req -new -key pikachu2.pem -out pikachu2.req
Using configuration from /usr/lib/ssl/openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [CZ]:
State or Province Name (full name) []:
Locality Name (eg, city) []:Breclav
Organization Name (eg, company) [Moje Firma, s.r.o.]:FIRMA a.s.
Organizational Unit Name (eg, section) []:IT Departement
Common Name (eg, YOUR name) []:pikachu2.firma.cz
Email Address []:sprava.site@firma.cz
```

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
mpressca@yoda:~$ openssl x509 -days 365 -signkey demoCA/private/cakey.pem \
                             -in pikachu2.req -req -out pikachu2.crt
Signature ok
subject=/C=CZ/L=Breclav/O=FIRMA a.s./OU=IT Departement/CN=pikachu2.moravi\
apress.cz/Email=sprava.site@firma.cz
Getting Private key
Enter PEM pass phrase:********  heslo certifikační autority
mpressca@yoda:~$ ls -l pikachu2.*
-rw-r--r--    1 mpressca mpressca     1001 Dec 14 16:08 pikachu2.crt
-rw-r--r--    1 mpressca mpressca      887 Dec 14 16:04 pikachu2.pem
-rw-r--r--    1 mpressca mpressca      733 Dec 14 16:06 pikachu2.req
mpressca@yoda:~$
```

Význam jednotlivých souborů je následující:

`pikachu.crt`

Obsahuje veřejný klíč ve formátu PEM

`pikachu.pem`

Obsahuje soukromý klíč klíč ve formátu PEM

`pikachu.req`

FIXME:

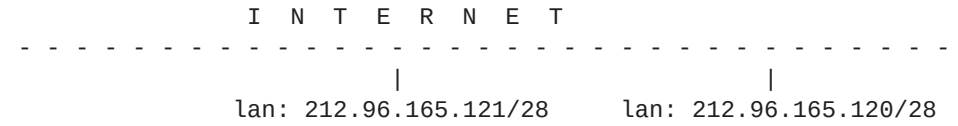
27.2.6. Zkušební konfigurace IPsec

* *$Header: /home/radek/cvs/unix-book/input/unix/sec-ipsec.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $*
* *section id="zkusebni-konfigurace-ipsec" xreflabel="Zkušební konfigurace IPsec" condition="author"*
* *print:$* ***psselect -p 165-176 unix.ps|foldprn -s12***

Testovací konfigurace sestává ze dvou internetových adres: `212.96.165.122` a `212.96.165.121` které přiděluji třem strojům.

- `pikachu` — představuje Linux server a používá adresu `212.96.165.122`
- `yoghurt` — představuje Linux klienta a používá adresu `212.96.165.121`
- `jirka` — představuje MSWindows klienta a používá adresu `212.96.165.121`
- `trada` — představuje MSWindows klienta a používá adresu `212.96.165.120`

Obrázek 27-5. Testovací konfigurace pro IPsec



```
eth1: 212.96.165.122/28    gprs: 160.218.177.191      gprs: 160.218.179.137
     +---------+              +---------------+             +-------+
     | pikachu |              | jirka/yoghurt |             | trada |
     +---------+              +---------------+             +-------+
eth0: 10.16.66.53/19       virt: 10.225.64.2/19       virt: 10.225.64.3/19
virt: 10.225.64.1/19
          |
        L A N
```

Nastavení síťových rozhraní rozepsané do jednotlivých parametrů jsou:

	Server	jirka	trada
ip	`212.96.165.122`	`212.96.165.121`	`212.96.165.120`
netmask	`255.255.255.240`		
network	`212.96.165.112`		
broadcast	`212.96.165.127`		
gateway	`212.96.165.126`		
nameserver	`212.96.165.113`		

27.2.6.1. První pokus

Testovací konfigurace `pikachu 212.96.165.122 yoghurt 212.96.165.121`

Postupuji podle IPSEC: secure IP over the Internet (http://lartc.org/howto/lartc.ipsec.html), a začaj jsem nejjednodušší konfigurací. Na obou strojích je Debian GNU/Linux Woody s podporou IPsec. Konfiguraci ukládám zatím pouze do skriptů. Na straně „serveru“ `pikachu` jsem si tedy vytvořil soubor definující IPsec spojení a spustil jej.

Příklad 27-5. `pikachu:ipsec1.conf`

```
#!/usr/sbin/setkey -f
# pikachu:/etc/racoon/ipsec.conf
# Spojeni s yoghurt

# Flush all
flush;
spdflush;

# Security Association
add 212.96.165.121 212.96.165.122 ah 24500 -A hmac-md5 "1234567890123456";
add 212.96.165.121 212.96.165.122 esp 24501 -E 3des-cbc "123456789012123456789012";

# NO Secury Policy
```

Na straně druhého počítače „klienta“ jsem vytvořil obdobný soubor

Příklad 27-6. yoghurt:ipsec1.conf

```
#!/usr/sbin/setkey -f
# yoghurt:/etc/racoon/ipsec.conf
# Spojeni s pikachu
```

```
# Clean
flush;
spdflush;

add 212.96.165.121 212.96.165.122 ah 24500 -A hmac-md5 "1234567890123456";
add 212.96.165.121 212.96.165.122 esp 24501 -E 3des-cbc "123456789012123456789012";

# Secure Policy
spdadd 212.96.165.121 212.96.165.122 any -P out ipsec
        esp/transport//require
        ah/transport//require;
```

Po startu skriptu i na straně klienta jsem spustil ping

```
$ ping 212.96.165.122
```

a ve výpisu **tcpdump** se objevili řádky

```
# tcpdump -i eth0
15:30:05.835737 121.firma.cz > 122.firma.cz: AH(spi=0x00005fb4,seq=0xd2): ESP(spi=0x00005fb5,seq=0xd2) (DF)
15:30:05.837341 122.firma.cz > 121.firma.cz: icmp: echo reply
15:30:06.835704 121.firma.cz > 122.firma.cz: AH(spi=0x00005fb4,seq=0xd3): ESP(spi=0x00005fb5,seq=0xd3) (DF)
15:30:06.837299 122.firma.cz > 121.firma.cz: icmp: echo reply
15:30:07.835691 121.firma.cz > 122.firma.cz: AH(spi=0x00005fb4,seq=0xd4): ESP(spi=0x00005fb5,seq=0xd4) (DF)
15:30:07.837286 122.firma.cz > 121.firma.cz: icmp: echo reply
15:30:08.835684 121.firma.cz > 122.firma.cz: AH(spi=0x00005fb4,seq=0xd5): ESP(spi=0x00005fb5,seq=0xd5) (DF)
.
.
```

27.2.6.2. Zabezpečení jednoduchého spojení

FIXME:

Příklad 27-7. yoghurt:ipsec2.conf

```
#!/usr/sbin/setkey -f
# yoghurt:/etc/racoon/ipsec.conf
# Spojeni s pikachu

# Flush all
flush;
spdflush;

# Security Association (SA) {source destination instruction} for AH
add 212.96.165.122 212.96.165.121 ah 15700 -A hmac-md5 "1234567890123456";
add 212.96.165.121 212.96.165.122 ah 24500 -A hmac-md5 "1234567890123456";

# Security Association for ESP
add 212.96.165.122 212.96.165.121 esp 15701 -E 3des-cbc "123456789012123456789012";
add 212.96.165.121 212.96.165.122 esp 24501 -E 3des-cbc "123456789012123456789012";

# Secure Policy
spdadd 212.96.165.121 212.96.165.122 any -P out ipsec
        esp/transport//require
        ah/transport//require;
spdadd 212.96.165.122 212.96.165.121 any -P in ipsec
        esp/transport//require
        ah/transport//require;
```

27.2.6.3. Automatická výměna klíčů

Pro tento účel jsem již použil démona `racoon`. Jeho konfigurace, uložená v souboru `/etc/racoon/racoon.conf` je stejná pro oba stroje:

```
# /etc/racoon/racoon.conf

log debug2;
path pre_shared_key "/etc/racoon/psk.txt";

remote anonymous
{
        exchange_mode main;
        doi ipsec_doi;
        situation identity_only;

        my_identifier address;

        lifetime time 2 min;
        initial_contact on;
        proposal_check obey;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group 2;
        }
}

sainfo anonymous
{
        pfs_group 1;
        lifetime time 2 min;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Dále jsem definoval sdílené tajemství. Tím bylo heslo `tajneheslo`. Na „serveru“ `pikachu` je uloženo v souboru `/etc/racoon/psk.txt`

```
# IPv4/v6 addresses
212.96.165.121  tajneheslo
# USER_FQDN
# FQDN
```

Na straně klienta je pak ve stejném souboru uvedeno

```
# IPv4/v6 addresses
212.96.165.122  tajneheslo
# USER_FQDN
# FQDN
```

Tato konfigurace není úplná, ještě bylo třeba doplnit skripty spd.conf. Na Serveru:

```
#!/usr/sbin/setkey -f
```

```
flush;
spdflush;

spdadd 212.96.165.122 212.96.165.121 any -P out ipsec
        esp/transport//require
        ah/transport//require;
spdadd 212.96.165.121 212.96.165.122 any -P in ipsec
        esp/transport//require
        ah/transport//require;
```

a na klientovi

```
#!/usr/sbin/setkey -f

flush;
spdflush;

spdadd 212.96.165.121 212.96.165.122 any -P out ipsec
        esp/transport//require
        ah/transport//require;
spdadd 212.96.165.122 212.96.165.121 any -P in ipsec
        esp/transport//require
        ah/transport//require;
```

Tyto skripty se spustí na obou stranách ještě před startem démona **racoon**.

27.2.6.4. Testování proti MSWindows

Dalším krokem bylo vyzkoušet IPsec proti MSWindows. Soubor `racoon.conf` jsem upravil a nahradil některé číselné konstanty symbolickými, aby bylo jasné co se má nastavit na MSWindows klientu.

```
log debug2;

path pre_shared_key "/etc/racoon/psk.txt";

remote anonymous
{
        exchange_mode main;
        doi ipsec_doi;
        situation identity_only;

        my_identifier address;

        lifetime time 4 min;
        initial_contact on;
        proposal_check obey;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp768;
        }
}

sainfo anonymous
```

```
{
        pfs_group modp768;
        lifetime time 4 min;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Rovněž jsem msel upravit skript `spd.conf`, neb zkoušený klient nějak nechodil s AH.

```
#!/usr/sbin/setkey -f

flush;
spdflush;

spdadd 212.96.165.122 212.96.165.121 any -P out ipsec
        esp/transport//require;
#       ah/transport//require;
spdadd 212.96.165.121 212.96.165.122 any -P in ipsec
        esp/transport//require;
#       ah/transport//require;
```

27.2.6.5. Vytvoření certifikátů

Další vecí kterou si odzkoušíme je autentizace obou stran pomocí certifikátů. Proto si ale nejdříve zmíněné certifikáty musíme vytvořit. Postupujeme podle návodu [xref linkend="openssl.cert"/]. Pokud nemáme certifikační autoritu, nebo si je nechceme pro tento účel vytvářet, můžeme certifikáty podepsat sebou místo certifikační autoritou. T.j. každý certifikát je podepsán sam sebou. Popíšeme si případ s certifikační autoritou.

FIXME: Vytvoříme si certifikační autoritu. Výsledkem jsou soubory:

- `cacert.pem` — veřejný certifikát
- `crl.pem` — ?*revokační seznam*?

FIXME: Poté vytvoříme certifikát pro server `pikachu`. Výsledkem budou soubory:

- `pikachu.key` — soukromý klíč, zůstává jen na serveru
- `pikachu.pem` — veřejný certifikát, publikujeme
- `pikachu.p12` — *pro MSWindows*

FIXME: A na konec vytvoříme certifikát pro klienta `jirkanb`. Výsledkem jsou soubory: Výsledkem jsou soubory:

- `jirkanb.key` — soukromý klíč, zůstává jen na klientovi
- `jirkanb.pem` — veřejný certifikát, publikujeme
- `jirkanb.p12` — *pro MSWindows*

A ještě jeden certifikát pro dalšího klienta.

```
mpressca@yoda:~$ CA -newreq
```

```
⋮
Enter PEM pass phrase:********     ←──heslo k vytvářenému certifikátu
Verifying password - Enter PEM pass phrase:******** ←──a ještě jednou
⋮
Country Name (2 letter code) [CZ]:CZ
State or Province Name (full name) []:.
Locality Name (eg, city) []:Breclav
Organization Name (eg, company) [Moje Firma, s.r.o.]:FIRMA a.s.
Organizational Unit Name (eg, section) []:IT Departement
Common Name (eg, YOUR name) []:trada.firma.cz
Email Address []:vladislav.tvaruzek@firma.cz
⋮
A challenge password []:<Enter>
An optional company name []:<Enter>
Request (and private key) is in newreq.pem
mpressca@yoda:~$ CA -sign
⋮
Enter PEM pass phrase:******** heslo ke klíči certifikační autority
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'CZ'
localityName          :PRINTABLE:'Breclav'
organizationName      :PRINTABLE:'FIRMA a.s.'
organizationalUnitName:PRINTABLE:'IT Departement'
commonName            :PRINTABLE:'trada.firma.cz'
emailAddress          :IA5STRING:'vladislav.tvaruzek@firma.cz'
Certificate is to be certified until Dec 13 14:16:27 2005 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
⋮
Signed certificate is in newcert.pem
mpressca@yoda:~$ mv newreq.pem trada.key
mpressca@yoda:~$ mv newcert.pem trada.pem
mpressca@yoda:~$ openssl pkcs12 -export -in trada.pem -inkey trada.key \
                      -certfile demoCA/cacert.pem -out trada.p12
Enter PEM pass phrase:********  heslo k certifikátu
Enter Export Password:<Enter>
Verifying password - Enter Export Password:<Enter>
mpressca@yoda:~$
```

Tím máme všechny klíče vytvořeny. Teď je umístníme na server a klienta a opravíme konfiguraci. Na server umístníme soubory: `cacert.pem`, `crl.pem`, `pikachu.key`, ?`pikachu.pem`?, `jirkanb.pem`, `trada.pem`.

```
pikachu:/etc/racoon# ls -l certs
total 28
-rw-r--r--    1 root     root         1281 Dec 13 13:47 cacert.pem
-rw-r--r--    1 root     root          516 Dec 13 13:47 crl.pem
-rw-r--r--    1 root     root         3702 Dec 13 13:47 jirkanb.pem
-rw-------    1 root     root         1696 Dec 13 13:47 pikachu.key
-rw-------    1 root     root         2941 Dec 13 13:47 pikachu.p12
-rw-r--r--    1 root     root         3702 Dec 13 13:47 pikachu.pem
```

```
-rw-r--r--    1 root     root         3711 Dec 13 15:20 trada.pem
pikachu:/etc/racoon#
```

Na klienta zase soubory: `cacert.pem`, `crl.pem`, `jirkanb.key`, ?`jirkanb.pem`?, `pikachu.pem`.

Na klienta `trada` uložíme soubory: `cacert.pem`, `crl.pem`, `trada.key`, ?`trada.pem`?, `pikachu.pem`.

Pro potřeby VPN budeme ještě potřebovat

```
mpressca@yoda:~$ openssl x509 -in demoCA/cacert.pem -noout -subject
subject= /C=CZ/L=Breclav/O=FIRMA a.s./OU=IT Departement/CN=Radek Hnilica/Email=ca@firma.cz
mpressca@yoda:~$
```

27.2.6.6. Připojení MS WindowsXP s použitím openssl certifikátů

Zdroje a odkazy:

- IPsec Tunneling Between FreeBSD Hosts (http://www.onlamp.com/pub/a/bsd/2001/12/10/ipsec.hmtl)
- Setting up a FreeBSD IPSec Tunnel (http://www.freebsddiary.org/ipsec-tunnel.php) by John J.Rushford Jr
- FreeBSD IPsec mini-HOWTO (http://www.daemonnews.org/200101/ipsec-howto.html)
- . ()

Vytvoření certifikátu pro server.

```
mpressca@yoda:~$ openssl req -new -keyout pikachu4.key -out pikachu4.req -days 360
mpressca@yoda:~$ cat pikachu4.req pikachu4.key >new.pem
mpressca@yoda:~$ mpressca@yoda:~$ openssl ca -policy policy_anything -out pikachu.crt -config

mpressca@yoda:~$ openssl req -new -x509 -keyout pikachu5.key -out pikachu5.req
```

Vytvoření klíče a žádosti o podpis.

```
mpressca@yoda:~$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyform PEM -keyout privkey.p
```

Podepsání žádosti certifikační autoritou

```
mpressca@yoda:~$ CA -sign
mpressca@yoda:~$ mv privkey.pem pikachu6.key
mpressca@yoda:~$ mv newreq.pem pikachu6.req
mpressca@yoda:~$ mv newcert.pem pikachu6.cert
```

Nyní pokročíme. Budeme zkoušet IPsec v režimu transport s použitím certifikátů. Certifikáty máme připraveny a nakopírovány na server i klienta. Upravíme tedy konfiguraci v souboru `racoon.conf`.

```
# Konfigurace s pouzitim certifikatu

log debug2;

path certificate "/etc/racoon/certs";

remote anonymous
{
```

```
        exchange_mode main;
        my_identifier asn1dn;
        peers_identifier asn1dn;
        lifetime time 4 min;

        certificate_type x509 "pikachu3.crt" "pikachu3.key";
        peers_certfile "jirkanb.pem";

        proposal {
                encryption_algorithm des;
                hash_algorithm sha1;
                authentication_method rsasig;
                dh_group modp768;
        }
}

sainfo anonymous {
        pfs_group modp768;
        lifetime time 4 min;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Skrip pro inicializaci SA databáze v souboru `/etc/racoon/spd.conf`:

```
#!/usr/sbin/setkey -f
# pikachu:/etc/racoon/ipsec.conf
# Spojeni s yoghurt,jirkanb,trada

### Flush all
flush;
spdflush;

### Security Policy

### jirka .firma.cz
# TRANSPORT MODE
#spdadd 212.96.165.122 212.96.165.121 any -P out ipsec
#        esp/transport//require;
#spdadd 212.96.165.121 212.96.165.122 any -P in ipsec
#        esp/transport//require;

# TUNNEL MODE
spdadd 10.16.64.0/19 10.225.64.0/19 any -P out ipsec
        esp/tunnel/10.16.66.53-10.225.64.3/require;
spdadd 10.225.64.0/19 10.16.64.0/19 any -P in ipsec
        esp/tunnel/10.225.64.3-10.16.66.53/require;



### trada.firma.cz from gprs: 160.218.179.137
spdadd 212.96.165.122 160.218.179.137 any -P out ipsec
        esp/transport//require;
spdadd 160.218.179.137 212.96.165.122 any -P in ipsec
        esp/transport//require;
```

```
# trada.firma.cz from lan: 212.96.165.120/28
spdadd 212.96.165.122 212.96.165.120 any -P out ipsec
        esp/transport//require;
spdadd 212.96.165.120 212.96.165.122 any -P in ipsec
        esp/transport//require;
```

Při vytváření části popisující tunel k jitkovi jsem vycházel z Setting up a FreeBSD IPSec Tunnel (http://www.freebsddiary.org/ipsec-tunnel.php) by John J.Rushford Jr. Obecně se tunel vytvoří takto:

```
spdadd my_local_net peer_local_net any -P out ipsec
        esp/tunnel/my_local_ip-peer_local_ip/require;
spdadd peer_local_net my_local_net any -P in ipsec
        esp/tunnel/peer_local_ip-my_local_ip/require;
```

Podle jiného dokumentu to má být ovšem jinak:

```
spdadd my_local_net peer_local_net any -P out ipsec
        esp/transport/my_public_ip-peer_public_ip/require;
spdadd peer_local_net my_local_net any -P in ipsec
        esp/transport/peer_public_ip-my_public_ip/require;
```

Vzor pro nastavování *Security Policy* pro transport je:

```
spdadd my_public_ip peer_public_ip any -P out ipsec
        esp/transport//require;
spdadd peer_public_ip my_public_ip any -P in ipsec
        esp/transport//require;
```

a nebo podle FreeBSD IPsec mini-HOWTO (http://www.daemonnews.org/200101/ipsec-howto.html), kde stroj A má adresu 1.2.3.4 a stroj B zase 5.6.7.8, takto

```
#!/bin/sh
setkey -FP
setkey -F

setkey -c << EOF
spdadd 1.2.3.4/32 5.6.7.8/32 any -P out ipsec
 esp/transport/1.2.3.4-5.6.7.8/require;
spdadd 5.6.7.8/32 1.2.3.4/32 any -P in ipsec
 esp/transport/5.6.7.8-1.2.3.4/require;
EOF
```

Podle dokumentace k programu **setkey** je struktura příkazu **spdadd** následující: `spdadd [-46n] src_range dst_range upperspec policy;` Kde *`src_range`* a *`dst_range`* jsou ve tvaru

- address
- address/prefixlen
- address[port]
- address/prefixlen[port]

Parametr *`upperspec`* pak může být:

- any — stands for any protocol
- ip4

- icmp6

Policy:

- -P *direction* discard
- -P *direction* nine
- -P *direction* ipspec protocol/mode/src-dst/level [...]

A restartujeme:

```
pikachu:/etc/racoon# /etc/init.d/racoon stop
Stopping IKE (ISAKMP/Oakley) server: racoon.
pikachu:/etc/racoon# ./spd.conf
pikachu:/etc/racoon# /etc/init.d/racoon start
Starting IKE (ISAKMP/Oakley) server: racoon.
pikachu:/etc/racoon#
```

Tentokrát je na straně MS WindowsXP použita vnitřní implementace IPsec. Při konfiguraci WindowsXP byly použity stránky Nata Carlsona (http://www.natecarlson.com/linux/ipsec-x509.php). Potřebný program **ipsec** je stažený z vpn.ebootis.de (http://vpn.ebootis.de/package.zip).

Na WinXP je potřeba doinstalovat z orig. CD WinXP z adresáře `\support\tools\setup.exe` kompletní instalace. Po té je potřeba zkopírovat z adresáře `c:\program files\support tools\ipseccmd.exe` do adresáře, kde je rozpakovany IPsec od Nata Carlsona.

Výsledný konfigurační soubor `ipsec.conf` je:

```
conn roadwarrior
 left=%any
        right=212.96.165.122
 rightca="C=CZ, L=Breclav, O=FIRMA a.s., OU=IT Departement, \
CN=Radek Hnilica, E=ca@firma.cz"
 network=auto
 auto=start
 pfs=yes

conn roadwarrior-net
 left=%any
        right=212.96.165.122
        rightsubnet=212.96.165.122/28
 rightca="C=CZ, L=Breclav, O=FIRMA a.s., OU=IT Departement, \
CN=Radek Hnilica, E=ca@firma.cz"
 network=auto
 auto=start
 pfs=yes
```

Nepodaří se navázat spojení. Na straně serveru se objeví v deníku následující:

```
Dec 14 14:04:27 pikachu racoon: DEBUG: send packet from 212.96.165.122[500]
Dec 14 14:04:27 pikachu racoon: DEBUG: send packet to 160.218.179.137[500]
Dec 14 14:04:27 pikachu racoon: DEBUG: src4 212.96.165.122[500]
Dec 14 14:04:27 pikachu racoon: DEBUG: dst4 160.218.179.137[500]
Dec 14 14:04:27 pikachu racoon: DEBUG: 1 times of 148 bytes message will be sent to 212.96.165.122[500]
Dec 14 14:04:27 pikachu racoon: DEBUG:  bb437566 a5ca8302 e8006166 4db9a1d6 04100200 00000000 00000094 \
  0a000064 38e171f6 8837b555 28f437df 55e58b16 30953439 bc348592 4911417f d006de3e d111aa90 115d118e e48cc89f \
  55b76d12 4ee957bd 832e0875 ddfaccb6 87190c0c f34990d2 661e9710 5345b7c1 e8ce2e74 bcadf9c9 ecfccf1e 00c49324 \
  6df89f9c 00000014 845ac279 b40650f2 c3554622 2751b278
Dec 14 14:04:27 pikachu racoon: DEBUG: resend phase1 packet bb437566a5ca8302:e80061664db9a1d6
Dec 14 14:04:37 pikachu racoon: ERROR: phase1 negotiation failed due to time up. bb437566a5ca8302:e80061664db9a1d6
.
.
Dec 14 15:04:50 pikachu racoon: DEBUG: filename: /usr/etc/racoon/certs/trada.pem
Dec 14 15:04:50 pikachu racoon: ERROR: failed to get peers CERT.
Dec 14 15:04:54 pikachu racoon: DEBUG: 148 bytes from 212.96.165.122[500] to 212.96.165.120[500]
```

```
.
.
Dec 14 15:15:37 pikachu racoon: ERROR: unknown Informational exchange received.
.
.
Dec 14 15:25:20 pikachu racoon: DEBUG: malformed cookie received or the spi expired.
Dec 14 15:25:52 pikachu racoon: DEBUG: ===
Dec 14 15:25:52 pikachu racoon: DEBUG: 84 bytes message received from 212.96.165.120[500] to 212.96.165.122[500]
Dec 14 15:25:52 pikachu racoon: DEBUG:  0f89de27 bb035ad4 09aabda3 fe6aa8d5 08100501 04827136 00000054 600f9521 \
    aaa09a54 d5f65e7d 3a8752f9 a8c5dd2e 948b6444 d810c16e 68947437 cbf9b6bd 1ca7dd14 00009b0d 8a673d10 8b5cba1b \
    efcc7735
Dec 14 15:25:52 pikachu racoon: ERROR: unknown Informational exchange received.
.
.
Dec 14 15:26:34 pikachu racoon: DEBUG: filename: /etc/racoon/certs/pikachu.key
Dec 14 15:26:34 pikachu racoon: ERROR: failed to get private key.
Dec 14 15:26:34 pikachu racoon: ERROR: failed to process packet.
Dec 14 15:26:34 pikachu racoon: ERROR: phase1 negotiation failed.
.
.
Dec 14 16:22:34 pikachu racoon: DEBUG: filename: /etc/racoon/certs/pikachu2.pem
Dec 14 16:22:34 pikachu racoon: ERROR: failed to get my CERT.
Dec 14 16:22:34 pikachu racoon: ERROR: failed to get own CERT.
Dec 14 16:22:34 pikachu racoon: ERROR: failed get my ID
Dec 14 16:22:34 pikachu racoon: ERROR: failed to process packet.
Dec 14 16:22:34 pikachu racoon: ERROR: phase1 negotiation failed.
.
.
Dec 16 14:25:56 pikachu racoon: DEBUG: suitable outbound SP found: 212.96.165.122/32[0] 212.96.165.121/32[0] proto=any dir=out.
Dec 16 14:25:56 pikachu racoon: DEBUG: sub:0xbffffb40: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=in
Dec 16 14:25:56 pikachu racoon: DEBUG: db :0x809bf90: 212.96.165.122/32[0] 212.96.165.121/32[0] proto=any dir=out
Dec 16 14:25:56 pikachu racoon: DEBUG: sub:0xbffffb40: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=in
Dec 16 14:25:56 pikachu racoon: DEBUG: db :0x80a0f20: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=out
Dec 16 14:25:56 pikachu racoon: DEBUG: sub:0xbffffb40: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=in
Dec 16 14:25:56 pikachu racoon: DEBUG: db :0x80a1220: 212.96.165.122/32[0] 160.218.179.137/32[0] proto=any dir=out
Dec 16 14:25:56 pikachu racoon: DEBUG: sub:0xbffffb40: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=in
Dec 16 14:25:56 pikachu racoon: DEBUG: db :0x809f398: 160.218.179.137/32[0] 212.96.165.122/32[0] proto=any dir=out
Dec 16 14:25:56 pikachu racoon: DEBUG: sub:0xbffffb40: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=in
Dec 16 14:25:56 pikachu racoon: DEBUG: db :0x809f770: 212.96.165.122/32[0] 212.96.165.120/32[0] proto=any dir=out
Dec 16 14:25:56 pikachu racoon: DEBUG: sub:0xbffffb40: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=in
Dec 16 14:25:56 pikachu racoon: DEBUG: db :0x809f9a8: 212.96.165.120/32[0] 212.96.165.122/32[0] proto=any dir=out
Dec 16 14:25:56 pikachu racoon: NOTIFY: no in-bound policy found: 212.96.165.121/32[0] 212.96.165.122/32[0] proto=any dir=in
Dec 16 14:25:56 pikachu racoon: DEBUG: new acquire 212.96.165.122/32[0] 212.96.165.121/32[0] proto=any dir=out
Dec 16 14:25:56 pikachu racoon: ERROR: failed to get sainfo.
.
.
Dec 20 13:01:58 pikachu racoon: ERROR: rejected enctype: DB(prop#1:trns#1):Peer(prop#1:trns#4) = 3DES-CBC:DES-CBC
Dec 20 13:01:58 pikachu racoon: ERROR: rejected authmethod: DB(prop#1:trns#1):Peer(prop#1:trns#4) = pre-shared key:RSA signatures
Dec 20 13:01:58 pikachu racoon: ERROR: rejected hashtype: DB(prop#1:trns#1):Peer(prop#1:trns#4) = SHA:MD5
Dec 20 13:01:58 pikachu racoon: ERROR: no suitable proposal found.
Dec 20 13:01:58 pikachu racoon: ERROR: failed to get valid proposal.
Dec 20 13:01:58 pikachu racoon: ERROR: failed to process packet.
```

27.2.6.7. Další pokus s WinXP

Zdroje a odkazy:

- PDF dokument IPsec Tunnel Mode Between Windows XP Professional and OpenBSD with X.509v3 Certificate Authentication (http://www.cs.umd.edu/users/mvanopts/xp2obsd.pdf)
- IPSec Architecture (http://www.microsoft.com/technet/itsolutions/network/security/ipsecarc.mspx)
- IPSec Resource for Windows 2000 (http://labmice.techtarget.com/networking/ipsec.htm)
- . ()
- . ()

Vytvoření certifikátů provedem následujícím postupem. Začneme nejdříve vytvořením certifikační autority. Vytvoříme si tedy klíč `ca.key` a žádost o certifikát `ca.req`.

```
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyout ca.key -out ca.req
```

Poté žádost o certifikát podepíšeme vytvořeným klíčem. T.j. certifikát certifikační autority je podepsán klíčem té samé autority.

```
$ openssl x509 -req -days 9999 -in ca.req -signkey ca.key -out ca.crt
```

Tím máme hotovu certifikační autoritu. Jedná se o soubory: `ca.crt`, `ca.key`. Certifikát `ca.crt` bude distribuován na jednotlivé stanice/routery/server a klíč `ca.key` si pečlivě uschováme.

Nyní si vytvoříme certifikáty pro jednotlivé stroje. Uvádím příklad vytvoření certifikátu pro stroj `pikachu`. Vytvoříme klíč a žádost o certifikát

```
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyout pikachu.key \
          -out pikachu.req
```

Žádost o certifikát `pikachu.req` podepíšeme certifikační autoritou a získáme certifikát stroje `pikachu.crt`.

```
$ openssl x509 -req -days 370 -in pikachu.req  -out pikachu.crt \
          -CA ca.crt -CAkey ca.key -CAcreateserial
```

U dalších strrojů postupujeme obdobně. Pro stroje s Windows XP ještě zabalíme všechny požadované certifikáty do „pkcs 12“ balíčku.

```
$ openssl pkcs12 -export -in jirka.crt -inkey jirka.key -certfile ca.crt \
          -out jirka.p12
```

Vytvořený balíček certifikátů je v souboru `jirka.p12`. Tento přeneseme na stanici a naimportujeme.

* *Poznámky:* ***openssl req -neq -nodes -newkey rsa:1024 -sha1 -keyform PEM -keyout ca.key -outform PEM -out ca.req*** *Vytvoření klíče a žádosti o certifikát certifikační autority. Na Linuxovem routeru nakopirujeme pikachu.crt pikachu.key a ca.crt #* ***ln -s ca.crt $(openssl x509 -noout -hash -in ca.crt).0***

Jiný způsob podepisování certifikátů. Vyžaduje vytvořit prostředí certifikační autority. Žádost o certifikát autority podepíšeme klíčem autority.

```
$ openssl ca -policy policy_anything -out ca.crt -infiles ca.req
```

Na straně Linuxového serveru jsou nainstalovány balíčky

```
pikachu:/etc/racoon# dpkg -l ipsec-tools racoon
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name                    Version                 Description
+++-=======================-=======================-======================
ii  ipsec-tools             0.3.3-0.backports.org.1   IPsec tools for Linux
ii  racoon                  0.3.3-0.backports.org.1   IPSEC IKE keying daemon
```

Na straně serveru/routeru je konfigurace uložena v adresáři `/etc/racoon/`. Konfigurace na straně serveru v souboru `racoon.conf`.

```
# Konfigurace s pouzitim certifikatu

log debug2;

path certificate "/etc/racoon/certs";

listen {
        isakmp 212.96.165.122;
}
```

```
remote anonymous
{
        exchange_mode main,base,aggressive;
        generate_policy on;
        passive off;
        doi ipsec_doi;
        situation identity_only;

        lifetime time 24 hour;          # min/hour

        my_identifier asn1dn;
        peers_identifier asn1dn;
        certificate_type x509 "pikachu.crt" "pikachu.key";

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method rsasig;
                dh_group modp1024;
        }
}

sainfo anonymous {
        pfs_group modp1024;
        lifetime time 1 hour;           # min/hour
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Skrip pro inicializaci SA databáze v souboru `/etc/racoon/spd.conf`:

```
#!/usr/sbin/setkey -f
# pikachu:/etc/racoon/ipsec.conf
# Spojeni s yoghurt,jirkanb,trada

### Flush all
flush;
spdflush;

### Security Policy

### jirka .firma.cz
# TRANSPORT MODE
#spdadd 212.96.165.122 212.96.165.121 any -P out ipsec
#        esp/transport//require;
#spdadd 212.96.165.121 212.96.165.122 any -P in ipsec
#        esp/transport//require;

# TUNNEL MODE
spdadd 10.16.64.0/19 10.225.64.0/19 any -P out ipsec
        esp/tunnel/10.16.66.53-10.225.64.3/require;
spdadd 10.225.64.0/19 10.16.64.0/19 any -P in ipsec
        esp/tunnel/10.225.64.3-10.16.66.53/require;

spdadd 212.96.165.122 212.96.165.121 any -P out ipsec
```

```
        esp/tunnel/212.96.165.122-212.96.165.121/require;
spdadd 212.96.165.121 212.96.165.122 any -P in ipsec
        esp/tunnel/212.96.165.121-212.96.165.122/require;


# trada.firma.cz from gprs: 160.218.179.137
spdadd 212.96.165.122 160.218.179.137 any -P out ipsec
        esp/transport//require;
spdadd 160.218.179.137 212.96.165.122 any -P in ipsec
        esp/transport//require;

# trada.firma.cz from lan: 212.96.165.120/28
spdadd 212.96.165.122 212.96.165.120 any -P out ipsec
        esp/transport//require;
spdadd 212.96.165.120 212.96.165.122 any -P in ipsec
        esp/transport//require;
```

A restartujeme:

```
pikachu:/etc/racoon# /etc/init.d/racoon stop
Stopping IKE (ISAKMP/Oakley) server: racoon.
pikachu:/etc/racoon# ./spd.conf
pikachu:/etc/racoon# /etc/init.d/racoon start
Starting IKE (ISAKMP/Oakley) server: racoon.
pikachu:/etc/racoon#
```

FIXME: Popsat konfiguraci na straně Windows XP.

Konfigurace je v souboru `.`

```
conn roadwarrior-net
        left=%any
        right=212.96.165.122
        rightsubnet=10.0.0.0/8
        rightca="C=CZ, L=Breclav, O=FIRMA a.s., OU=IT Departement, \
                 CN=pikachu.firma.cz, E=sprava.site@firma.cz"
        network=auto
        auto=start
        pfs=yes
```

Nepodaří se navázat spojení. Na straně serveru se objeví v deníku následující:

```
Dec 21 09:21:50 pikachu racoon: DEBUG: anonymous configuration selected for 212.96.165.121[500].
Dec 21 09:21:50 pikachu racoon: DEBUG: ===
Dec 21 09:21:50 pikachu racoon: INFO: respond new phase 1 negotiation: 212.96.165.122[500]<=>212.96.165.121[500]
Dec 21 09:21:50 pikachu racoon: INFO: begin Identity Protection mode.
.
.
.
Dec 21 09:42:58 pikachu racoon: DEBUG: resend phase1 packet b09491a5d9e18873:73020e8c7209fb6b
Dec 21 09:42:58 pikachu racoon: DEBUG: ===
Dec 21 09:42:58 pikachu racoon: DEBUG: 56 bytes message received from 212.96.165.121[500] to 212.96.165.122[500]
Dec 21 09:42:58 pikachu racoon: DEBUG:  b09491a5 d9e18873 73020e8c 7209fb6b 0b100500 4eab84e5 00000038 0000001c \
    00000001 01100004 b09491a5 d9e18873 73020e8c 7209fb6b
Dec 21 09:42:58 pikachu racoon: DEBUG: receive Information.
Dec 21 09:42:58 pikachu racoon: ERROR: ignore information because the message has no hash payload.
Dec 21 09:43:08 pikachu racoon: DEBUG: 153 bytes from 212.96.165.122[500] to 212.96.165.121[500]
Dec 21 09:43:08 pikachu racoon: DEBUG: sockname 212.96.165.122[500]
Dec 21 09:43:08 pikachu racoon: DEBUG: send packet from 212.96.165.122[500]
Dec 21 09:43:08 pikachu racoon: DEBUG: send packet to 212.96.165.121[500]
Dec 21 09:43:08 pikachu racoon: DEBUG: src4 212.96.165.122[500]
Dec 21 09:43:08 pikachu racoon: DEBUG: dst4 212.96.165.121[500]
```

```
Dec 21 09:43:08 pikachu racoon: DEBUG: 1 times of 153 bytes message will be sent to 212.96.165.122[500]
.
.
Dec 21 09:43:08 pikachu racoon: DEBUG: receive Information.
Dec 21 09:43:08 pikachu racoon: ERROR: ignore information because the message has no hash payload.
Dec 21 09:43:18 pikachu racoon: ERROR: phase1 negotiation failed due to time up. b09491a5d9e18873:73020e8c7209fb6b
```

Uvedenou chybu

```
Dec 21 09:42:58 pikachu racoon: ERROR: ignore information because the message has no hash payload.
```

popisují

- Zpráva (racoon 387) message has no hash payload (http://www.kame.net/racoon/racoon-ml/msg00367.html) a následující reakce
- VPN between Check Point VPN-1 NG and Linux 2.6.x (IKE daemon racoon) (http://www.fw-1.de/aerasec/ng/vpn-racoon/CP-VPN1-NG-Linux-racoon.html)
- Phase 1 schlägt fehl (https://www.spenneberg.com/1050.html)
- racoon: ERROR: because the message has no hash payload. (https://www.spenneberg.com/1346.html)
- racoon mit x509 zur Checkpoint FW-1 (https://www.spenneberg.com/2125.html)
- racoon: verifity_cert on, keine Verbindung (https://www.spenneberg.com/2132.html)
- Linux als Roadwarrior mit racoon (http://www.spenneberg.com/2292.html)
- This is the secret Racoon error message decoder ring. (http://www.fefe.de/racoon.txt)
- . ()

27.2.7. Popis některých balíčků

* *deb:ipsec-tools;*

27.3. L2TP

Odkazy

- Home of l2tpd, the Layer 2 Tunneling Protocol Daemon (http://www.l2tpd.org)
- Using a Linux L2TP/IPsec VPN server with Windows 2000/XP (http://www.jacco2.dds.nl/networking/win2000xp-freeswan.html)
- Using a Linux L2TP/IPsec VPN server (http://www.jacco2.dds.nl/networking/freeswan-l2tp.html)
- Using a Linux server with the Microsoft L2TP/IPSec VPN Client (http://www.jacco2.dds.nl/networking/msl2tp.html)
- Using a Linux server with third-party L2TP/IPsec clients (http://www.jacco2.dds.nl/networking/l2tp3rdparty.html)
- PDF dokument L2TP Roadwarrior Guidebook (http://docs.astaro.org/howtos/L2TP_Roadwarrior_Guidebook.pdf)
- IPSec v kernelu 2.6 část I (http://www.abclinuxu.cz/clanky/show/59641?varianta=print) a II (http://www.abclinuxu.cz/clanky/show/60983?varianta=print)
- Linux Kernel 2.5/2.6 using KAME-tools (http://www.ipsec-howto.org/x237.html)
- OpenSSL Based PKI Implementation in Real World :: A Cookbook (http://www.rajeevnet.com/crypto/ca/ca-paper.html)

- Linux Kernel 2.6 using KAME-tools (NAT-Traversal) (http://www.ipsec-howto.org/x265.html)

l2tpd je program jenž realizuje klientskou i serverovou část konce l2tp spojení. Domovskou stránka projektu je na l2tpd.org (http://www.l2tpd.org/). Ke dnešnímu dni (2005-01-04) je ke stažení verze 0.69 (http://www.l2tpd.org/download.html). V archivech Debianu se nachází verze 0.67-1.2 (stable) (http://packages.qa.debian.org/l/l2tpd.html) a 0.70-pre20031121-2 (unstable) (http://packages.qa.debian.org/l/l2tpd.html). V stable (`Sarge`) v tuto chvíli žádná verze není.

27.3.1. Překlad ze zdrojů

Po přečtení řady informací jako *IPSec v kernelu 2.6 část I (http://www.abclinuxu.cz/clanky/show/59641?varianta=print) a II (http://www.abclinuxu.cz/clanky/show/60983?varianta=print)* jsem dospěl k názoru že si l2tpd preventívně přeložím. Použil jsem virtuální server bone jenž mám k tomuto účelu vytvořen. Takže po obnovení čistého serveru jsem upravil zdroje v `/etc/apt/sources.list` a přidal jsem tam řádky:

```
deb-src http://localhost:9999/main/ stable main non-free contrib
deb-src http://localhost:9999/main/ sarge main non-free contrib
deb-src http://localhost:9999/main/ sid main non-free contrib
```

Poté jsem přistoupil k instalování programů:

```
bone:/usr/src# apt-get install dpkg-dev debhelper gcc libc-dev
```

A závěrečné stažení zdrojů:

```
bone:/usr/src# apt-get source -t sarge l2tpd
Reading Package Lists... Done
Building Dependency Tree... Done
Need to get 150kB of source archives.
Get:1 http://localhost sid/main l2tpd 0.70-pre20031121-2 (dsc) [589B]
Get:2 http://localhost sid/main l2tpd 0.70-pre20031121-2 (tar) [127kB]
Get:3 http://localhost sid/main l2tpd 0.70-pre20031121-2 (diff) [22.2kB]
Fetched 150kB in 56s (2662B/s)
dpkg-source: extracting l2tpd in l2tpd-0.70-pre20031121
```

Nyní jsem si v klidu prohlédl zdroje programu l2tpd, prozkoumal závislosti a vůbec se v něm porozhlédl. Takže ze závislostí jsem narazil na

```
Build-Depends: debhelper (>= 3.0.5)
```

Poté jsem l2tpd jednoduše přeložil:

```
bone:/usr/src# apt-get source -t sarge --build l2tpd
bone:/usr/src# cd l2tpd-0.70-pre20031121/
bone:/usr/src# dpkg-buildpackage -b -uc
```

Poznámka: Ve skutečnosti jsem se pokusil nejdříve stáhnout a přeložit zdroje l2tpd. A podle chybových hlášek doinstlovával potřebné balíčky s programy a knihovnami. Nezapomeňte že virtuální server `bone` je čistě nainstalovaný s minimem programů.

Protokol o překladu:

```
bone:/usr/src# apt-get source -t sarge --build l2tpd
Reading Package Lists... Done
Building Dependency Tree... Done
Need to get 150kB of source archives.
```

```
Get:1 http://localhost sid/main l2tpd 0.70-pre20031121-2 (dsc) [589B]
Get:2 http://localhost sid/main l2tpd 0.70-pre20031121-2 (tar) [127kB]
Get:3 http://localhost sid/main l2tpd 0.70-pre20031121-2 (diff) [22.2kB]
Fetched 150kB in 56s (2661B/s)
Skipping unpack of already unpacked source in l2tpd-0.70-pre20031121
dpkg-buildpackage: source package is l2tpd
dpkg-buildpackage: source version is 0.70-pre20031121-2
dpkg-buildpackage: source maintainer is Jean-Francois Dive <jef@debian.org>
dpkg-buildpackage: host architecture is i386
 debian/rules clean
dh_testdir
dh_testroot
rm -f build-stamp
/usr/bin/make clean
make[1]: Entering directory ‘/usr/src/l2tpd-0.70-pre20031121’
rm -f l2tpd.o pty.o misc.o control.o avp.o call.o network.o avpsend.o scheduler.o file.o aaa.o md5.o l2tpd
make[1]: Leaving directory ‘/usr/src/l2tpd-0.70-pre20031121’
dh_clean
 debian/rules build
dh_testdir
/usr/bin/make
make[1]: Entering directory ‘/usr/src/l2tpd-0.70-pre20031121’
cc -ggdb -Wall -DDEBUG_PPPD -DDEBUG_CONTROL -DDEBUG_ENTROPY -Wall -DSANITY -DLINUX  -DIP_ALLOCATION     -c -o l2t
pd.o l2tpd.c
cc -ggdb -Wall -DDEBUG_PPPD -DDEBUG_CONTROL -DDEBUG_ENTROPY -Wall -DSANITY -DLINUX  -DIP_ALLOCATION     -c -o pty
.o pty.c
cc -ggdb -Wall -DDEBUG_PPPD -DDEBUG_CONTROL -DDEBUG_ENTROPY -Wall -DSANITY -DLINUX  -DIP_ALLOCATION     -c -o mis
c.o misc.c
cc -ggdb -Wall -DDEBUG_PPPD -DDEBUG_CONTROL -DDEBUG_ENTROPY -Wall -DSANITY -DLINUX  -DIP_ALLOCATION     -c -o con
trol.o control.c
control.c: In function ‘handle_packet’:
control.c:1660: warning: unused variable ‘tv’
cc -ggdb -Wall -DDEBUG_PPPD -DDEBUG_CONTROL -DDEBUG_ENTROPY -Wall -DSANITY -DLINUX  -DIP_ALLOCATION     -c -o avp
.o avp.c
avp.c: In function ‘validate_gen_avp’:
avp.c:303: parse error before ‘int’
avp.c:306: ‘i’ undeclared (first use in this function)
avp.c:306: (Each undeclared identifier is reported only once
avp.c:306: for each function it appears in.)
avp.c:308: ‘found’ undeclared (first use in this function)
make[1]: *** [avp.o] Error 1
make[1]: Leaving directory ‘/usr/src/l2tpd-0.70-pre20031121’
make: *** [build-stamp] Error 2
Build command ’cd l2tpd-0.70-pre20031121 && dpkg-buildpackage -b -uc’ failed.
E: Child process failed
bone:/usr/src#
```

Kritická část souboru `avp.c`

```
300 int validate_gen_avp(int attr,  struct tunnel *t, struct call *c,
301                                          void *data, int datalen) {
302         (void)data; (void)datalen;
303         int i = 0, found = 0;
304
305     if(t->sanity) {
306                 for(i = 0; i < 8; i++) {
307                         if(c->msgtype == avps[attr].allowed_states[i])
308                                 found++;
309                 }
310                 if(!found)
311                         return -EINVAL;
312         }
313         return 0;
314 }
```

Letmým pohledem na kód se domnívám že překladači gcc verze 2.95.4-14 se asi nelíbí konstrukce na řádku 302. Protože se dále nikde oba parametry nepožívají, tak jsem celý řádek zakomentoval. Překlad pak proběhl úspěšně.

27.3.2. Instalace a konfigurace

* *rcsinfo=$Header: /home/radek/cvs/unix-book/input/unix/sec-l2tp.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $*

FIXME:

27.3.3. L2TP přes IPsec

* *rcsinfo=$Header: /home/radek/cvs/unix-book/input/unix/sec-l2tp.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $*

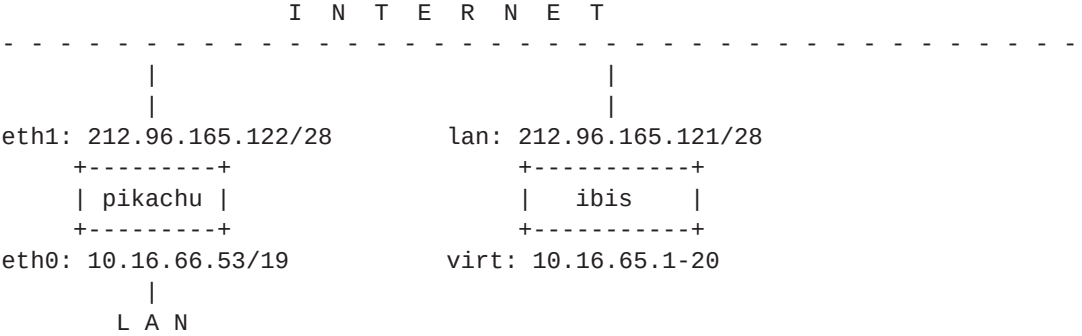
FIXME:

27.3.3.1. Linux server a Mac OS-X klient

* *rcsinfo=$Header: /home/radek/cvs/unix-book/input/unix/sec-l2tp.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $*

FIXME:

Obrázek 27-6. Obrázek konfigurace Linux server — Mac OS-X roadwarrior



Začneme serverem. Na tom je nainstalován Racoon a jeho konfigurace v souboru `/etc/racoon/racoon.conf` je:

```
# pikachu:/etc/racoon/racoon.conf

log debug;      #notify/debug/debug2
path pre_shared_key "/etc/racoon/psk.txt";

listen {
        isakmp 212.96.165.122;
}

remote anonymous
{
        exchange_mode main;
        generate_policy on;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}
```

```
sainfo anonymous {
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Sekce `remote anonymous` může být nastavena jako `remote 212.96.165.121`.

Soubor `/etc/racoon/psk.txt` jenž obsahuje sdílená tajemství:

```
# pikachu:/etc/racoon/psk.txt
# IPv4/v6 addresses
212.96.165.121  hloupeheslo
```

Databázi politik inicializuju souborem `/etc/racoon/ipsec.start`

```
#!/bin/sh
# pikachu:/etc/racoon/ipsec.start

/usr/sbin/setkey -c <<EOF

### Flush all
flush;
spdflush;

spdadd 212.96.165.122[1701] 0.0.0.0/0 any -P out ipsec esp/transport//require;

EOF
```

Skript `/etc/racoon/ipsec.start` spouštíme buď to ručně, nebo si jeho spouštění zakomponujeme do skriptu `/etc/init.d/racoon` jak jsem to udělal já. Tím máme zabezpečeno že po restartu routeru bude korektně nakonfigurován i VPN server. Tím jsme dokončili konfiguraci IPsec vrstvy a můžeme přikročit k dalšímu.

Dalším krokem je konfigurace l2tpd. Ta se nachází v souboru `/etc/l2tpd/l2tpd.conf`:

```
; pikachu:/etc/l2tpd/l2tpd.conf

; Nastaveni serveru
[lns default]                           ; Our fallthrough LNS definition
ip range = 10.16.65.1-10.16.65.20       ; * Allocate from this IP range
local ip = 10.16.66.53                  ; * Our local IP to use
require chap = yes                      ; * Require CHAP auth. by peer
refuse pap = yes                        ; * Refuse PAP authentication
require authentication = yes            ; * Require peer to authenticate
name = LinuxVPNserver                   ; * Report this as our hostname
ppp debug = yes                         ; * Turn on PPP debugging
pppoptfile = /etc/ppp/options.l2tpd     ; * ppp options file
```

Konfigurace pro ppp je v souboru `/etc/ppp/options.l2tpd`.

```
ipcp-accept-local
ipcp-accept-remote
ms-dns  10.16.66.18
ms-dns  10.16.66.19
noccp
auth
crtscts
idle 1800
```

```
mtu 1410
mru 1410
nodefaultroute
debug
lock
deflate 9
proxyarp
connect-delay 5000
```

Ještě nadefinujeme účet pro klienta s názvem `ibis` a heslem `spojme`.

```
# pikachu:/etc/ppp/chap-secrets
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
ibis            *       spojme                  *
```

Server nakonfigurovaný máme, teď nakonfigurujeme klienta.

FIXME: tady bych měl uvést obrázky jednotlivých dialogů. Než to udělám tak je popíši aspoň slovy.

VPN (L2TP) Moraviapress Configuration

Description: Moraviapress
Server Address: 212.96.165.122
Account Name: ibis —→*/etc/ppp/chap-secrets*
Authentication: Use Password: spojme —→*/etc/ppp/chap-secrets*
Shared Secret: hloupeheslo —→*/etc/racoon/psk.txt*

Připojení z klienta `ibis` proběhne úspěšně a jsem schopen komunikovat s počitači uvnitř firemní sítě.

Program **l2tpd** vypíše protokol:

```
ourtid = 54705, entropy_buf = d5b1
ourcid = 45983, entropy_buf = b39f
check_control: control, cid = 0, Ns = 0, Nr = 0
handle_avps: handling avp's for tunnel 54705, call 45983
message_type_avp: message type 1 (Start-Control-Connection-Request)
protocol_version_avp: peer is using version 1, revision 0.
framing_caps_avp: supported peer frames:async sync
hostname_avp: peer reports hostname ”
assigned_tunnel_avp: using peer's tunnel 16
receive_window_size_avp: peer wants RWS of 4.  Will use flow control.
check_control: control, cid = 0, Ns = 1, Nr = 1
handle_avps: handling avp's for tunnel 54705, call 45983
message_type_avp: message type 3 (Start-Control-Connection-Connected)
control_finish: Connection established to 212.96.165.121, 51646.  Local: 54705,\
 Remote: 16.  LNS session is 'default'
check_control: control, cid = 0, Ns = 2, Nr = 1
handle_avps: handling avp's for tunnel 54705, call 45983
message_type_avp: message type 10 (Incoming-Call-Request)
message_type_avp: new incoming call
ourcid = 54332, entropy_buf = d43c
assigned_session_avp: assigned session id: 23589
call_serno_avp: serial number is 1
check_control: control, cid = 23589, Ns = 3, Nr = 2
handle_avps: handling avp's for tunnel 54705, call 54332
message_type_avp: message type 12 (Incoming-Call-Connected)
tx_speed_avp: transmit baud rate is 1000000
frame_type_avp: peer uses:async frames
start_pppd: I'm running:  "/usr/sbin/pppd" "passive" "-detach"\
 "10.16.66.53:10.16.65.1" "refuse-pap" "auth" "require-chap" "name"\
 "LinuxVPNserver" "debug" "file" "/etc/ppp/options.l2tpd" "/dev/ttyp0"
```

```
control_finish: Call established with 212.96.165.121, Local: 54332,\
 Remote: 23589, Serial: 1
```

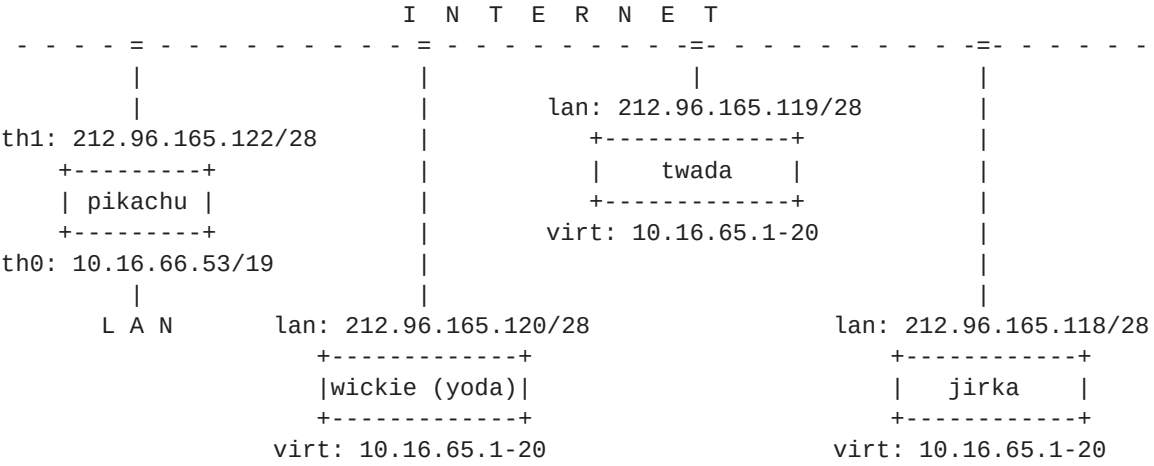
27.3.3.2. Linux server a MS Windows

Odkazy

- Using a Linux L2TP/IPsec VPN server with Windows 2000/XP (http://www.jacco2.dds.nl/networking/win2000xp-freeswan.html)
- . ()
- … ()

FIXME:

Obrázek 27-7. Obrázek konfigurace Linux server — WinXP roadwarrior



Při konfigurování jsem vyšel z konfigurace pro MAC OS-X klienta. Tuto jsem rozšířil, takže v současné době dovoluje server připojení jak MAC OS-X klienta tak WinXP.

27.3.3.2.1. Nastavení serveru s použitím předsdíleného klíče

V první fázi zkusíme použití předsdíleného klíče (sdíleného tajemství). Konfigurace je obdobná jako pro klienta s Mac OS-X.

V konfiguraci serveru racoon, v souboru `/etc/racoon/racoon.conf` jsem vytvořil sekce pro všechny stanice které se budou připojovat.

```
# pikachu:/etc/racoon/racoon.conf

log debug;      #notify/debug/debug2
path pre_shared_key "/etc/racoon/psk.txt";

listen {
        isakmp 212.96.165.122;
}

# Zkusebni klient Ibis
remote 212.96.165.121 {
        exchange_mode main;
        generate_policy on;
```

```
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}


# Zkusebni notas Twada
remote 212.96.165.119 {
        exchange_mode main;
        generate_policy on;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}


# Zkusebni notas Jirka
remote 212.96.165.118 {
        exchange_mode main;
        generate_policy on;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}


sainfo anonymous {
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Rovněž jsem pro každou stanici vytvořil řádek se sdíleným tajemstvím v souboru `/etc/racoon/psk.txt`

```
# pikachu:/etc/racoon/psk.txt
# IPv4/v6 addresses
212.96.165.121  totojetajne
212.96.165.119  silenypristup
212.96.165.118  hruzahrozna
```

Poslední co zbylo byly účty. Ty se vytváří v souboru `/etc/ppp/chap-secrets`

```
# pikachu:/etc/ppp/chap-secrets
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
ibis            *       mujklic                 *
```

```
twada           *       4blbost                 *
jirka           *       bububu                  *
```

27.3.3.2.2. Nastavení klienta `twada` se sdíleným tajemstvím

Jako další jsme odzkoušeli připojit místo MAC-OS X počítač s MS Windows XP. Na straně serveru jsem přidal do souboru `/etc/racoon/racoon.conf` *remote* sekci pro klienta `twada` jemuž jsem přidělil pevnou adresu 212.96.165.119.

```
remote 212.96.165.119 {
        exchange_mode main;
        generate_policy on;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}
```

Doplnil jsem pro nej sdílené tajemství do souboru `/etc/racoon/psk.txt`, a vytvořil účet s heslem v souboru `/etc/ppp/chap-secrets`.

Na straně klienta s MS Windows XP je konfigurace:

FIXME: popsat konfiguraci.

27.3.3.2.3. Použití klíčů asymetrické kryptografie

FIXME:

Začneme vytvářením klíčů

FIXME: Vytvoříme si klíč pro Win klienta. Takto učiníme ve třech krocích. Prvním je vytvoření žádosti o certifikát `wickie.crt`.

```
$ cd ~/firma/mpress/ca
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyout wickie.key \
          -out wickie.req
Using configuration from /usr/lib/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.......++++++
.............++++++
writing new private key to 'wickie'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [CZ]:
State or Province Name (full name) []:.
```

```
Locality Name (eg, city) []:Breclav
Organization Name (eg, company) [Moje Firma, s.r.o.]:FIRMA a.s.
Organizational Unit Name (eg, section) []:IT Departement
Common Name (eg, YOUR name) []:wickie
Email Address []:radek.hnilica@moraviapress.cz

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Vytvořenou žádost o certifikát `wickie.req` podepíšeme certifikátem autority `ca.crt` a vytvoříme tak certifikát `wickie.crt`

```
$ openssl x509 -req -days 370 -in wickie.req -out wickie.crt \
          -CA ca.crt -CAkey ca.key -CAcreateserial
Signature ok
subject=/C=CZ/L=Breclav/O=FIRMA a.s./OU=IT Departement/CN=wickie/Email=ra\
dek.hnilica@moraviapress.cz
Getting CA Private Key
```

Pro potřeby MS Windows klienta musíme vytvořit balíček klíčů `wickie.p12`.

```
$ openssl pkcs12 -export -in wickie.crt -inkey wickie.key \
          -certfile ca.crt -out wickie.p12
Enter Export Password:
Verifying password - Enter Export Password:
$
```

Podle uvedénoh vzoru vytvoříme klíče i pro ostatní stanice.

```
$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyout jirka.key \
          -out jirka.req
$ openssl x509 -req -days 370 -in jirka.req -out jirka.crt \
          -CA ca.crt -CAkey ca.key -CAcreateserial
Signature ok
subject=/C=CZ/L=Breclav/O=FIRMA a.s./OU=IT Departement/CN=jirka/Email=sp\
rava.site@moraviapress.cz
Getting CA Private Key
$ openssl pkcs12 -export -in jirka.crt -inkey jirka.key \
          -certfile ca.crt -out jirka.p12

$ openssl req -new -nodes -newkey rsa:1024 -sha1 -keyout twada.key -out twada.req
$ openssl x509 -req -days 370 -in twada.req -out twada.crt \
          -CA ca.crt -CAkey ca.key -CAcreateserial
Signature ok
subject=/C=CZ/L=Breclav/O=FIRMA a.s./OU=IT Departement/CN=twada/Email=sprava.site@moraviapress
Getting CA Private Key
$ openssl pkcs12 -export -in twada.crt -inkey twada.key \
          -certfile ca.crt -out twada.p12
```

Takto vytvořené certifikáty ale ve WinXP nefungují. Měl by fungovat následující postup

```
Client certificate
openssl req -new -keyout newreq.pem -out newreq.pem -days 730
openssl ca -policy policy_anything -out newcert.pem -extensions xpclient_ext -extfile xpextens
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out cert-clt.p12 -clcerts
openssl pkcs12 -in cert-clt.p12 -out cert-clt.pem
```

```
openssl x509 -inform PEM -outform DER -in cert-clt.pem -out cert-clt.der
```

```
$ openssl req -new -keyout twada2.pem -out twada.pem -days 370
```

```
$ openssl rsa -in passworded-key.pem -out /etc/ipsec.d/private/freeswan-fw.key
$ openssl rsa -in newreq.pem -out twada.key
```

27.3.4. Nezapracované materiály

Jednoduchá ukázka konfiguračního souboru `/etc/l2tp/l2tpd.conf` převzatá z Using a Linux L2TP/IPsec VPN server (http://www.jacco2.dds.nl/networking/freeswan-l2tp.html):

```
; Sample l2tpd.conf
;
[global]
; listen-addr = 192.168.1.98

[lns default]
ip range = 192.168.1.128-192.168.1.254
local ip = 192.168.1.99
require chap = yes
refuse pap = yes
require authentication = yes
name = LinuxVPNserver
ppp debug = yes
pppoptfile = /etc/ppp/options.l2tpd
length bit = yes
```

27.4. OpenVPN

Zdroje a odkazy:

- OpenVPN (http://openvpn.net)
- **FIXME:** ()

27.4.1. Instalace

Ke dnešnímu dni 2005-02-07 je v Debianu k dispozici balíček OpenVPN pouze v Sarge a Sidu. Jedná se o verze `1.99+2-rc6` (Sarge) a `1.99+2-rc12` (Sid). Pro praktické použití na routeru s Woody jsem tedy šáhl do Debian Backports (Backports.ORG) (http://www.backports.org/) kde je verze `openvpn_1.99+2.beta17-0.backports.org.1`.

Do apt zdrojů přidáme odkaz na Debian Backports (Backports.ORG) (http://www.backports.org/). V mém případě jdu přes firemní apt-proxy.

```
deb http://debian:9999/backports woody openvpn
```

Jinou možností je využít systému `rootstuff` a nasměrovat apt do něj:

```
deb file:/root/stuff/deb/woody/openvpn
```

Máme-li apt připraveno a aktualizováno, nainstalujeme OpenVPN. V mém případě ukazuji instalaci na router `pikachu`.

```
pikachu:~# apt-get install openvpn
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  liblzo1
The following NEW packages will be installed:
  liblzo1 openvpn
0 packages upgraded, 2 newly installed, 0 to remove and 0  not upgraded.
Need to get 280kB of archives. After unpacking 918kB will be used.
Do you want to continue? [Y/n] y
Get:1 http://debian woody/main liblzo1 1.07-1 [40.9kB]
Get:2 http://debian woody/openvpn openvpn 1.99+2.beta17-0.backports.org.1 [239kB
]
Fetched 280kB in 14s (19.9kB/s)
Preconfiguring packages ...
Selecting previously deselected package liblzo1.
(Reading database ... 7852 files and directories currently installed.)
Unpacking liblzo1 (from .../liblzo1_1.07-1_i386.deb) ...
Selecting previously deselected package openvpn.
Unpacking openvpn (from .../openvpn_1.99+2.beta17-0.backports.org.1_i386.deb) ..
.
Setting up liblzo1 (1.07-1) ...

Setting up openvpn (1.99+2.beta17-0.backports.org.1) ...
Stopping openvpn:.
Starting openvpn:.
```

Program máme nainstalován a zbývá nám jeho konfigurace.

27.4.2. Vytvoření CA a správa klíčů

K provozu OpenVPN potřebujeme openssl certifikáty. Ty si můžeme vytvořit podle (openssl.cert), nebo použijeme skripty dodané s balíčkem OpenVPN jak popíši dále.

V adresáří `/usr/share/doc/openvpn/examples/easy-rsa` najdeme skripty ukázkové certifikační autority. Překopíroval jsem si tedy celý adresář k sobě do `~/firma/tatofirma/ovca`. Tímto způsobem mohu udržovat více cartifikačních autorit pro různé firmy či různé použití. Součástí ukázkové CA je velmi dobrý popis v soubor `README.gz`. Následující postup z tohoto popisu vychází.

Upravíme hodnoty v souboru s proměnnýma `vars` tak aby odpovídali našim potřebám. Poté vytvoříme certifikační autoritu.

```
$ . vars
$ ./clean-all
$ ./build-ca
```

Vytvoření klíče/certifikátu pro stroj je jednoduché:

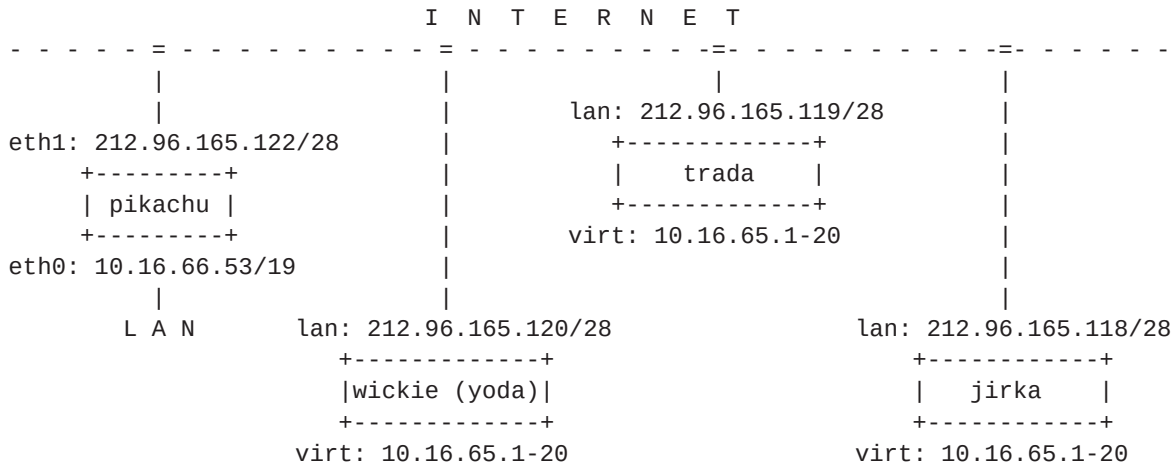
```
$ ./build-req stroj
$ ./sign-req stroj
```

Získáme soubory *`stroj`*`.crt` a *`stroj`*`.key`. V prvním je veřejný certifikát a v druhém soukromý klíč.

27.4.3. Konfigurace server více klientů

Jako vždycky, uvedu obrázek konfigurace.

Obrázek 27-8. Obrázek konfigurace Linux server — WinXP roadwarrior



František mi laskavě poskytnul svou konfiguraci. Františkův `/etc/openvpn/vpn_server.conf`:

```
# Konfigurak pro openvpn v rezimu server na linux stroji

mode server
tls-server
dev tap0
ifconfig 192.168.200.1 255.255.255.0
ifconfig-pool 192.168.200.10 192.168.200.20 255.255.255.0
#port 5001
proto udp
duplicate-cn
push "route 192.168.1.0 255.255.255.0"
ca /etc/openvpn/cacert.pem
cert /etc/openvpn/cert.pem
key /etc/openvpn/key.pem
dh /etc/openvpn/dh1024.pem

log-append /var/log/openvpn
status /var/run/openvpn/vpn.status 10
user openvpn
group openvpn
comp-lzo
verb 3
```

27.4.3.1. Konfigurce serveru

Aktuální konfigurace na serveru:

```
# Configuration file for server - many Road Wariors

local 216.95.167.112
```

```
port 1194
proto udp
dev tun         # tun/tap

tls-server
mode server

ca      /etc/openvpn/ca.crt             # Root CA
cert    /etc/openvpn/pikachu.crt        # My Public Certificate
key     /etc/openvpn/pikachu.key        # My Secret Key
dh      /etc/openvpn/dh1024.pem         # Diffie Hellman parameters.

server 10.225.65.0 255.255.255.0
#ifconfig 10.225.65.1 10.225.65.2
#ifconfig-pool 10.225.65.10 10.225.65.30        # IP range clients
#ifconfig-pool-persist ipp.txt

comp-lzo

user nobody
group nogroup

persist-key
persist-tun

push            "route 10.0.0.0 255.0.0.0"      # route to company network
push            "dhcp-option DNS 10.17.64.17"
push            "dhcp-option DOMAIN example.cz"
push            "redirect-gateway"

# Client specific configuration
client-config-dir ccd

# Create logs
status          /var/log/openvpn-status.log
log-append      /var/log/openvpn.log
verb 3
```

K rozlišení klientů mezi sebou slouží příkaz `client-config-dir` který má jako parametr název podadresáře s klientskými nastaveními. Klientské konfigurace jsou uloženy v samostatných souborech, jejichž jméno odpovídá CN v certifikátu. Například Jirka který má vystaven certifikát s `.../CN=jirka/...`, proto se soubor se jeho specifickým nastavením bude jmenovat `jirka`. V tomto souboru mu pak přidělím pevnou adresu, případně provedu další nastavení specifická jen pro něj.

```
ifconfig-push 10.225.65.13 10.225.65.14
```

Z rozsáhlé konfigurace OpenVPN vyberu jako další zajímavou a užitečnou věc spouštění skriptů při sestavení a rozpojení spojení. Pro tyto účely jsou v konfiguračním souboru příkazy `client-connect` a `client-disconnect`. Ukážeme si to na jednoduchém skriptu který bude hlásit sestavená a ukončená spojení. Do konfiguračního souboru přidáme

```
client-connect /etc/openvpn/connect
```

a vytvoříme jednoduchý skript `/etc/openvpn/connect`

```
#!/bin/sh
EMAIL=spravce@firma.cz
```

```
mail $EMAIL -s "OpenVPN Connect report" <<EOF
$(date "+Y-%m-%d %T")
Klient s CN=$1 a IP=$2 se prave pripojil do VPN.
-- connect
EOF
```

27.5. Nezapracované materiály

27.5.1. FreeS/WAN

Linux FreeS/WAN je implementace IPSEC a IKE pod Linux

Poznámka: Projekt FreeS/WAN byl oficiálně ukončen.

Odkazy

- Linux FreeS/WAN (http://www.freeswan.org/)
- Notes on setting up tunneling using Debian 2.4 (Sid) and Kernels 2.4.x (http://www.thing.dyndns.org/debian/vpn.htm)
- Securing a wireless network point with Debian and FreeS/WAN (http://www.pseudorandom.co.uk/2003/debian/ipsec/)
- FREESWAN.ca (http://www.freeswan.ca/) — neoficiální FreeS/WAN site

27.5.2. ipip tunel

```
# ip tunnel add tunl0 mode ipip remote remote_IP local 192.168.2.1
```

Příklad 27-8. Příklad nastavení ipip tunelu

Máme dva systémy

```
System A  eth0:10.0.1.1  eth1:192.168.1.254
System B  eth0:10.0.2.1  eth1:192.168.2.254
```

Na systému A nastavíme

```
ip tunnel add tun0 mode ipip remote 10.0.2.1 local 192.168.1.254
ip addr add 192.168.1.254 dev tun0
ip link set tun0 up
ip route add 192.168.2.0/24 dev tun0
```

Na systému B nastavíme

```
ip tunnel add tun0 mode ipip remote 10.0.1.1 local 192.168.2.254
ip addr add 192.168.2.254 dev tun0
ip link set tun0 up
ip route add 192.168.1.0/24 dev tun0
```

Kapitola 28. MikroTik

Jedním z nástrojů které v síti můžeme použít je router MikroTik. Zde není kompletní popis ani konfigurace, zde jsou jen mé poznámky k nastavování a provozování MikroTiku

Odkazy:

- www.MikroTik.com (http://www.mikrotik.com)

28.1. Prvotní nastavení

Při prvním zapnutí provedeme počáteční nastavení. Nejdříve se přihlásíme jako uživatel `admin`, heslo je prázdné.

```
Mikrotik Login: admin
Password:
```

První věc kterou uděláme je že nastavíme heslo. Nechcem přeci aby nám někdo cizí „administroval“ náš důležitý router.

```
[admin@MikroTik] > /password
old password:
new password: nove-heslo
retype new password: nove-heslo
```

Dále nastavíme adresu na ethernetovém rozhraní

```
[admin@MikroTik] > /ip address add address=192.168.1.2/24 interface=ether1
```

V tuto chvíli se již můžeme k zařízení připojit přes ethernet.

28.2. Firmware upgrade

Firmware na MikroTiku upradujeme jednoduše tak, že pomocí přenosu souborů ftp nahrajeme do mikrotiku balíček s novou verzí. Tento se jmenuje nějak takto:

routers-*architektura-verze*.npk

Například do RB112 budu nahrávat v tuhle chvíli aktuální verzi v souboru `routing-test-3.13-mipsle.npk`.

```
$ lftp 192.168.1.2
lftp 192.168.1.2:~> user admin *heslo*
lftp admin@192.168.1.2:~> ls
lftp admin@192.168.1.2:/> put routeros-mipsle-3.13.npk
10180519 bytes transferred in 17 seconds (570.8K/s)
```

Po úspěšném nahrání balíčku mikrotik prostě restartujeme. Na terminálu uvídíme průběh instalace balíčku s novou verzí firmware.

```
[admin@MikroTik] > /system reboot
```

```
Reboot, yes? [y/N]: y
system will reboot shortly


Rebooting...
Stopping services...
installing routeros-mipsle-3.13 [##################
```

Příkazem `/system resource print` si ověříme verzi firmware. A příkazem `/system check-installation` ověříme celou instalaci.

28.3. /system default-configuration

Toto nastavení bylo přidáno ve verzi 3.0rc5. Tak jak jsem jej pochopil jedná se o skript který se vykonává v různých okažicích. Jeho aktuální podobu zjistíme příkazem

```
[admin@MikroTik] /system default-configuration print
  script: #| IP address 192.168.88.1/24 is on ether1
          #| ether1 is enabled

          :global action

          # these commands are executed after installation or configuration reset
          :if ($action = "apply") do={
              :delay 5s
              /ip address add address=192.168.88.1/24 interface=ether1 comment="default config
          }

          # these commands are executed if user requests to remove default configuration
          :if ($action = "revert") do={
              /ip address {
                  :local o [find address="192.168.88.1/24" interface="ether1" comment="default
                :if ([:len $o] != 0) do={ remove $o }
              }
          }
```

Skript nastavíme příkazem `/system default-configuration get script`

28.4. Údržba konfigurace

/system reset-configuration

Nastaví celý MikroTik do původního továrního nastavení, v četně administrátorského hesla.

28.5. Rozesílání konfigurací

Created Monday 23/06/2008

script pro zasílání automatických záloh od Jardy Pecha

```
# Pracovni promenne

:local sysname


# ziskani informaci z MK

:set sysname ([/system identity get name])

:set time ([/system clock get time])

:set date ([/system clock get date])

/system backup save name=zaloha-jmeno

:delay 15

/tool e-mail send from=pechj@netel.cz to=pechj@netel.cz server=SMTP subject=("Mikrotik " . $sysname . " Za-
loha") body=("Zaloha ze dne " . $date . " " . $time) file=zaloha-jmeno.backup

:delay 5

file remove zaloha-jmeno.backup
```

28.6. Nezapracované informace přetažené ze Zimu

MikroTiku
Created Monday 02/04/2007

```
> ip addr print

> interface wireless print


> ip firewall export


Export Import gonfigurace

export file=nazev-souboru
import nazev-souboru.mac


queue tree> print from=[find parent=skupina1-down]
queue tree> remove from=[find parent=skupina1-down]
```

http://www.mikrotik.com/testdocs/ros/2.9/ip/flow.php

Nastavení hesla

> /user set admin password=*HESLO*

28.7. Konfigurace maškarády

Odkazy:

-
-
-

```
;;; nasquerade network
chain=srcnat out-interface=internet src-address=192.168.1.0/24
action=masquerade

chain=srcnat out-interface=internet src-address=192.168.1.0/24
action=src-nat to-addresses=xx.xx.xx.xx to-ports=0-65535
```

Kapitola 29. Routery a switche CISCO

* *chapter id="cisco" condition="author"*

Pár příkazů:

sh run

Výpis aktuálně bežící/používané konfigurace.

sh conf

Výpis konfigurace uložené ve flash paměti.

conf term

Zadání konfigurace z konzoly.

write mem

Zapsání aktuálně běžící konfigurace do flash paměti.

Kapitola 30. Ethernet

Síťování na ethernetu (lokální síti)

* *chapter id="fragment.ethernet"*

Síťování na lokální síti s použitím technologie etherent. Základy technologiií ethernet, protokoly a použití ethernetu.

30.1. Historie a přehled

Dopsat

30.2. DHCP

Protokol DHCP řeší dynamické přidělování IP adres počítačům v návaznosti na MAC adresu jejich síťové karty.

DHCP server nainstalujeme z balíčku dhcp. Poté je třeba dhcpd nakonfigurovat. To učiníme editováním souboru `/etc/dhcpd.conf`. Před spuštěním je ještě potřeba povolit toto v souboru `/etc/init.d/dhcp`, řádek

```
run_dhcpd=0
```

přepíšeme na

```
run_dhcpd=1
```

30.3. BOOTP

SLuzbu BOOTP serveru poskytuji programy dhcp, bootp

30.4. Přepínače 3COM

Přepínač 3300TM 24 PT má vzdálenou správu. Jemožno použít webové rozhraní, nebo telnet

```
$ telnet sw33u1ozp

    login: admin
    password:
```

30.5. VLAN 802.1Q - Virtuální síťe na ethernetu

Odkazy a zdroje:

- LINUX VLAN + Cisco HOWTO (http://www.candelatech.com/~greear/vlan/howto.html)
- linux/VLAN/dhcp/iptables notes (http://www.planetconnect.com/vlan/)

- V Debianu jso k 2003-09-02 balíčky `stable vlan 1.5-2`, `testing vlan 1.6-1` a `unstable vlan 1.6-1.1` — *User mode programs to enable VLANs on your ethernet devices*

30.5.1. Instalace

Podporu pro VLAN je třeby mít zakompilovánu v jádře. Jedná se o položku menu Networking options⟶802.1Q VLAN Support v konfiguračním souboru uvedenou jako `CONFIG_VLAN_8021Q`

Kapitola 31. SNMP *Simple Network Management Protocol*

* *chapter id="snmp" xreflabel="SNMP" condition="author"*

Odkazy a zdroje:

- SNMP Version 3 (SNMPv3) (http://www.ibr.cs.tu-bs.de/projects/snmpv3/)
- Simple Network Management Protocol (http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm)
- Portable SNMP Agent (http://www.iniche.com/products2/hex13.shtml)
- Linux CMU SNMP Project (http://www.gaertner.de/snmp/welcome-3.7last.html)
- SNMP FAQ (http://www.faqs.org/faqs/snmp-faq/)
- SNMP (http://www.webopedia.com/TERM/S/SNMP.html)
- The Simple Times (http://www.simple-times.org/) HTML Archive (http://www.simple-times.org/pub/simple-times/issues/)
- SNMP - Simple Network Managment Protocol (http://www2.rad.com/networks/1995/snmp/snmp.htm) by Yoram Cohen
- Linux SNMP Network Management Tools (http://linas.org/linux/NMS.html)
- SNMP for the Public Community (http://www.wtcs.org/snmp4tpc/) by Williams Technology Consulting Services
- SMI Network Management Private Enterprise Codes (http://www.iana.org/assignments/enterprise-numbers) Prefix: iso.org.dod.internet.private.enterprise (1.3.6.1.4.1)
- http://net-snmp.sourceforge.net/tutorial/mrtg/ ()NET-SNMP Tutorial -- MRTG
- Lessons about SNMP (http://www.et.put.poznan.pl/snmp/main/mainmenu.html) SNMP tutorial
- SNMP / Network Management Software (http://www.simpleweb.org/software/)
- Monitoring a link with SNMP (http://www.midcoast.com/~jp/snmp.html) and learn some wimpy shell scripting too
- Simon's SNMP Hacks (http://www.switch.ch/misc/leinen/snmp/)
- SNMP Overview and SNMP Security (http://www.solarwinds.net/Tools/SNMP.htm)
- Package: snmp (4.2.3-2) (http://packages.debian.org/stable/net/snmp) in Debian/GNU Linux Woody
- SNMP for linux (http://www.topology.org/comms/snmp.html) — linux SNMP software, information about SNMP, standards, management information bases (MIBs)

SNMP protokol byl vyvinut v roce 1988.

31.1. Experimenty se SNMP

```
$ snmpwalk -v 1 entropy public
```

```
$ snmpget -v 1 entropy public interfaces
```

IV. Služby

Kapitola 32. Certifikáty a certifikační autorita

Odkazy:

- Certifikační autorita a OpenSSL (http://digiweb.ok.cvut.cz/bezpecnost/certifikacni-autorita-s-openssl)
- TinyCA2 (http://tinyca.sm-zone.net/)
- **Seriál Jak na OpenSSL publikovný na Root.CZ**
  - Jak na OpenSSL (http://www.root.cz/clanky/jak-na-openssl/)
  - Jak na OpenSSL II (http://www.root.cz/clanky/jak-na-openssl-2/)
  - Jak na OpenSSL 3 (http://www.root.cz/clanky/jak-na-openssl-3/)
- Ubuntu OpenSSL (https://help.ubuntu.com/community/OpenSSL)

* ***FIXME:***

32.1. Vytvoření CA pomocí skriptů v Lenny

Odkazy:

- Certificate Authority (CA) with OpenSSL (http://www.debian-administration.org/article/Certificate_Authority_CA_with_OpenSSL)

```
# mkdir /etc/ssl/ca
#
#
#
#
```

32.2. Vytvoření certifikační autority ručně s pomocí OpenSSL

Nejdříve si ukážeme jak si takovou certifikační autoritu sami vytvoříme jen s pomocí OpenSSL. Vše co potřebujeme je balíček s programem který se na Debian Etch jmenuje `openssl`. Pokud jej nemáme nainstalován, nainstlaujeme:

```
# aptitude install openssl
```

Nyní si vybereme kde budeme mít adresář s certifikační autoritou. Mnou uváděný postup provádí vytvoření a práci s certifikační autoritou pod uživatelem. Já jsem si pro ukázkovou certifikační autoritu vybral adresář `~/firma/ca`. Při tomhle postupu si jako užvatel můžeme udržovat několik certifikačních autorit. Vytvoříme tedy adresář a přepneme se do něj.

```
$ mkdir -p ~/firma/ca
$ cd ~/firma/ca
```

Nyní potřebujeme konfigurační soubor pro openssl. Protože si tento budeme upravovat pro každou certifikační autoritu, nakopírujeme si ukázkový soubor do právě vytvořeného adresáře. Jako ukázkový jsem použil ten standardně instalovaný s balíčkem `openssl` jenž se nachází v `/etc/ssl/openssl.cnf`.

```
$ cp /etc/ssl/openssl.cnf .
```

Konfigurační soubor si upravíme. Nejdříve nastavíme adresář v kterém budou všechny soubory. Tento bude `~/firma/ca/ca`. Parametr jímž tak činíme se jmenuje `dir` a nachází se v sekci `[ CA_default ]`.

```
[ CA_default ]
dir = ./ca
```

Tato změna úplě stačí pro funkčnost. Ale je lépe podívat se i na několik dalších parametrů a nastavit si je přesně podle potřeb konkrétní certifikační autority. Jedná se v první řadě o parametry v sekci `[ req_distinguished_name ]`, které popisují komu je certifikát vydáván. Protože předpokládám že certifikáty vydávate pro vlastní potřebu, nebo potřebu firmy v které pracujete, má smysl tyto předvyplnit.

```
[ req_distinguished_name ]
countryName_default = CZ
stateOrProvinceName_default = .
localityName_default = Praha
0.organizationName_default = Firma, s.r.o.
organizationUnitName_default = IT oddeleni
```

Další nastavení které považuji za užitečné je zmena *policy* v souvislosti s parametrem `stateOrProvinceName`. Tento je totiž vyžadován, a v našem prostředí nemá smysl. Proto jej nastavím na `optional`. Parametr se nachází v sekci `[ policy_match ]`

```
[ policy_match ]
stateOrProvinceName = optional
```

Z řady parametrů ještě uvedu jeden, a to `default_days` který určuje počet dní na které je certifikát vydávan. Tedy počet dní ode dneška po které je certifikát platný. Po vypršení této doby je nutné certifikát prodloužit. Tento parametr se nachází v sekci `[ CA_defaults ]`

```
default_days=3600
```

Můžeme dle vlastní potřeby modifikovat i další parametry, tady laskavého čtenáře odkáži na dokumentaci k programu OpenSSL.

Máme vytvořen a k vlastnímu obrazu uprave konfigurační soubor a nyní si vytvoříme potřebné adresáře. Je to jednak samotný adresář `ca` ve kterém jsou všechny soubory certifikační autority, tak ji jeho podadresáře.

```
$ mkdir ca
$ mkdir ca/newcerts
$ mkdir ca/private
$ chmod 0700 ca/private
```

Dále si vytvoříme index certifikátů a inicializujeme jejich počitadlo.

```
$ touch ca/index.txt
$ echo 01 >ca/serial
```

Rovněž nastavíme přístupová práva aby se k naší certifikační autoritě nedostal nikdo jiný. Znalost klíče by mu titiž umožnila vydávat falešné certifikáty.

```
$ chmod 0700 ~/firma/ca
$ chmod 0700 ca
```

Máme vytvořeny adresáře a vytvoříme samotnou certifikační autopritu.

```
$ cd ~/firma/ca
$ openssl req -config openssl.cnf -new -x509 -out cacert.pem -keyout cakey.pem -days 7500 -new
```

Při vytváření klíče certifikační autority jsme vyzváni k zadání přístupové fráze k tomuto klíči. Pomocí fráze je samotný klíč zašifrován, takže dostane-li se k němu nepovolaná posoba, nemůže jej bez znalosti fráze použít. Tuto frázi je ovšem nutno zadávat kdykoliv se klíč použije, tedy při podepisování každé žádosti o certifikát. Jsme li si jistí že se k našemu klíči nikod nedostane, a potřebujeme li podepisování automatizovat, můžem zašifrování klíče pomocí přístupové fráze vypnout přepínačem `-nodes`.

* ***FIXME:*** *Upozornil bych na použití přepínače `-des3` který zajistí že samotný klíč bude zašifrován pomocí přístupové fráze. Toto je důležité neb kdyby se někdo dostal k souboru s klíčem, potřebuje pro jeho použití ještě znát přístupovou frázi. Na druhou stranu to taky znamená, že kdykoliv budeme chtít naší autoritou podepsat připravenou žádost, musíme zadávat přístupovou frázi. Pokud jsme si zcela jisti, že se k našemu klíči nikdo nedostane a zadavání fráze je pro nás velká překážka pro automatické podepisování žádostí, odstraníme tento přepínač a použijeme místo něj jiný `-nodes`.*

Protože program **openssl** očekává konfigurační soubor na standardních místech jako je `/etc/ssl/openssl.cnf`, musíme mu oznámit že má použít náš. Tak učiníme parametrem `-config openssl.cnf`. Místo tohoto parametry je možné použít proměnnou prostředí `OPENSSL_CONF`. To s výhodou učiníme později až budeme práci s certifikáty zjodnodušovat vytvořením několik vlastních skriptů.

Dalším parametrem na který bych chtěl upozornit je `-days 7500`. Tento určuje dobu trvání certifikátu, tedy do kdy platí, v počtu dní ode dneška. V případě certifikátu certifikační autority zvolíme dostatečně velkou hodnotu. V uvedeném případě je to cca 20 let.

A poslední parametr který zmíním je `-newkey rsa:4096`. Tento určuje délku rsa klíče v bitech. Pro dnešní dobu je doporučované minimum 2048. Protože se jedná o klíč certifikační autoritiy, použiji raději klíč delší a to 4096 bitů dlouhý.

Vytvořené soubory umístníme na místo kde jsou očekávany a nastavíme jim přístupová práva.

```
$ mv cacert.pem ca/         # certifikát autority, veřejný soubor
$ chmod 0400 cakey.pem      # soukromý klíč autority, je neveřejný
$ mv cakey.pem ca/private   # a je očekáván v tomto adresáři
```

Tímto máme certifikační autoritu připravenu k použití, k podepisování žádostí o certifikáty. Jak se toto dělá, si popíšeme v následující části.

32.3. Vytvoření certifikátu podepsaného naší certifikační autoritou

Vytvoříme žádost o klíč:

```
$ openssl req -config openssl.cnf -new -des3 -out request.pem -keyout key.pem -days 1098 -newk
```

Který necháme podepsat certifikační autoritě

```
$ openssl ca -config openssl.cnf -in request.pem -out cert.pem
```

Pokud potřebujeme vytvořit certifikát pro www server, zadáme při vytváření žádosti do pole `Common Name` adresu tohoto serveru. Například `www.example.com`.

32.4. Jednoduché úkony

Zde popíši řadu jednoduchých úkonů.

32.4.1. Změna přístupové fráze ke klíči

Pokud nejsme spokojeni s přístupovo frází ke klíč, s její kvalitou či existení, můžeme ji změnit.

```
$ openssl rsa -des3 -in oldkey.pem -out newkey.pem
```

Příkaz přečte klíč, zeptá se nás na přístupovou frázi a poté klíč zapíše do nového souboru s novou přístupovou frází. K zapsání této nové fráze budeme vyzváni. Pokud chceme odstranit přístupovou frázi úplně, použijeme příkaz.

```
$ openssl rsa -in oldkey.pem -out newkey.pem
```

Takto si můžeme měnit přístupové fráze dle libosti či je odstranit úplně.

32.5. Různé poznámky

Vygenerování žádosti o privátní klíč.

Příklad 32-1. ca_req

```
openssl req -new -out cert.req -keyout cert.key -nodes
```

Podepsání žádosti, t.j. vytvoření certifikátu.

Příklad 32-2. ca_sign

```
#!/bin/sh
openssl ca -out cert.crt -verbose -infiles cert.req
cat cert.crt cert.key > cert.pem
chmod 600 cert.key cert.pem
```

Vytvořen souboru .p12 včetně privátního klíče vhodný pro import do poštovního klienta Outlook.

Příklad 32-3. ca_export_outlook

```
#!/bin/sh
openssl pkcs12 -export -in cert.crt -inkey cert.key -out cert.p12
```

Prodloužení platnosti certifikátu.

```
openssl x509 -x509toreq -in cert.pem -signkey key.pem -out cert.req
openssl ca -revoke cert.pem
```

32.5.1. Vytvoření žádosti o certifikát

```
$ openssl req -new -key privkey.pem -out cert.csr
```

Nyní pošleme `cert.csr` k podpisu certifikační autoritě. Pokud certifikační autorita neakceptuje soubory v PEM formátu, přidáme při vytváření žádosti přepínač `-outform` následovyný klíčovým slovem určujícím formát.

32.5.2. Vytvoření testovacího certifikátu

Testovací certifikát, podepsaný sebou samým vytvoříme:

```
$ openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
```

32.5.3. Vytvoření RSA klíče

```
$ openssl genrsa -des3 -out privkey.pem 2048
```

Budete vyzváni k zadání ochranného hesla klíče. Jestli nechcete klíč chránit heslem, nepoužívejte přepínač `-des3`. Číslo 2048 je délka klíče v bitech. Nepoužívejte délky menší než 2048 bitů. Je to z důvodu bezpečnosti.

32.5.4. Vytvoření DSA klíče

DSA klíče mohou být používány jen k podpisu.

```
$ openssl dsaparam -out dsaparam.pem 2048

$ openssl gendsa -des3 -out privkey.pem dsaparam.pem
```

Kapitola 33. Elektronická pošta (e-mail)

SMTP, IMAP, POP3, MTA, MDA, ...

* *chapter id="email"*

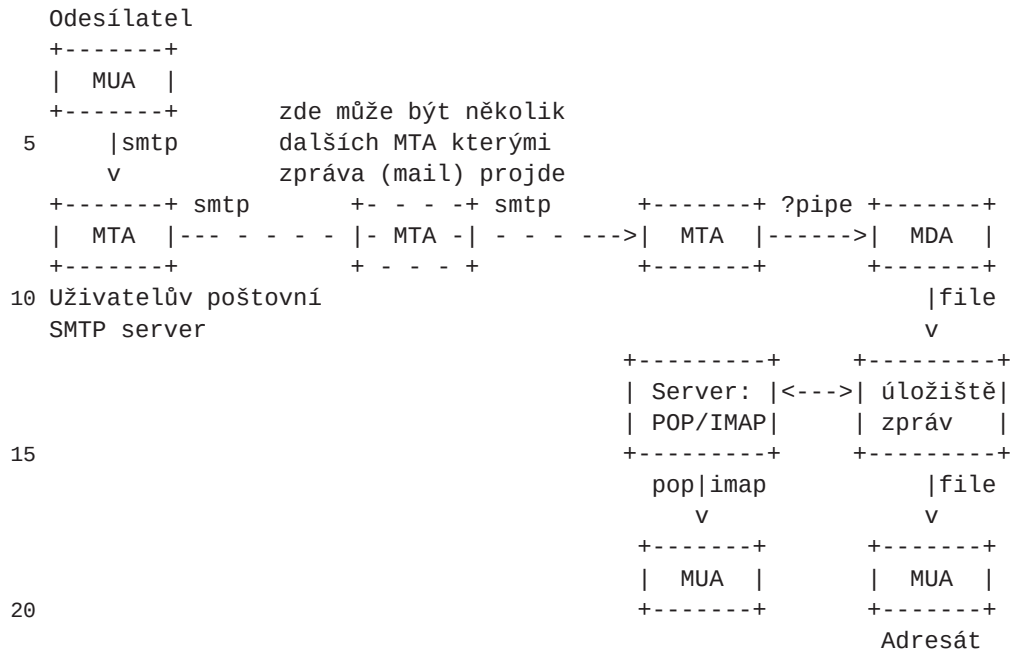
Vše kolem elektronické pošty. Poštovní servery, poštovní klienti, konfigurace, řešení typických úloh.

Elektronická pošta, email je jednou z nejstarších internetových služeb. Aby nedošlo k nedorozumění, elektronická pošta existovala již před internetem.

Tedy pod pojmem elektronická pošta (*email*) rozumíme řadu spolupracujících technologií které se starají o příjem, zpracování, směrování, doručení a předání elektronické pošty. Základem pošty je *SMTP* protokol pro přenos zpráv. Tímto protokolem spolu komunikují jednotlivé poštovní servery, MTA. Poštovním serverem, tedy MTA (*Message Transport Agent*) rozumíme program kterému můžeme protokolem SMTP poslat mail a on na základě hlavičky a vnitřního nastavní rozhodne jak s ním naloží. Může jej předat jinému MTA, který je „blíž“ adresátovi zprávy, nebo, je-li to náš domovský server (z hlediska pošty), předá zprávu k doručení programu který se jmenuje MDA (*Message Delivery Agent*). Následující obrázek se snaží podchytit životní cyklis emailu, a znázornit celou jeho cestu.

Obrázek 33-1. Život emailu

Obrázek pokoušející se znázornit celou cestu emailu od okamžiku vzniku až do přečtení adresátem.



Samotný email vzniká v programu který se nazývá MUA (*Message User Agent*) je to uživatelův oblíbený „poštovní“ porgram jako je třeba mail, mutt, Pegasus Mail, Fire Bird, ... V tomto programu tedy uživatel/odesílatel email napíše a odešle. MUA email pošle prvotnímu MTA což je nejčastěji lokální SMTP server. Bud’to běžící na stejném stroji nebo je to centrální SMTP server na síti či SMTP server u ISP. Tento email převezme a zajistí jeho směrování a přeposlání k dalšímu SMTP serveru který je „blíže“ adresáta.

V nejednodušším případě, kdy je uživatelův poštovní server totožný se poštovním serverem adresáta projde zpráva jen tímto jedním MTA. V běžném případě jsou to alespoň dva MTA kterými zpráva projde, SMTP server odesílatele a poštovní server příjemce.

Poslední MTA tedy cílový poštovní server jenž rozezná email jako „vlastní“ již tento nikam neposílá ale předá dalšímu programu jenž se nazývá MDA (*Message Delivery Agent*). Tento provede tzv. doručení, tedy operaci kdy email „vhodí“ do poštovní schránky (*mailboxu*) příjemce.

33.1. Poštovní protokoly

* *section id="email-protocols"*

Odkazy:

- Internet Message Access Protocol (IMAP) and Post Office Protocol (POP) (http://www.unet.univie.ac.at/aix/aixbman/commadmn/ml_imap_pop.htm)
- RFC2821 — Simple Mail Transfer Protocol (http://rfc.net/rfc2821.html)
- RFC2060 — Internet Message Access Protocol - Version 4 rev.1 (http://rfc.net/rfc2060.html)
- RFC1939 — Post Office Protocol - Version 32401 (http://rfc.net/rfc1939.html)

Protokoly jenž jsou používány k přenosu elektronické pošty v různých fázích. **FIXME:**popsat v kostce protokoly pop3, imap, smtp

33.1.1. Post Office Protocol (POP3)

* *section id="pop3"*

Protokol POP3 je jednoduchý protokol pro výběr pošty.

Odkazy a zdroje:

- POP3 Telnet Tutorial for Liquid Web Inc. customers (http://manual.liquidweb.com/chapter3/pop3.htm)

33.1.1.1. Zkoušení a testování

Potřebujeme-li odzkoušet že POP3 server je v pořádku a funguje, můžeme si zahrát na POP3 klienta. Pomocí programu **telnet** se přihlásíme a ručně zadáváme příkazy.

```
$ telnet pop3.firma.cz 110
```

poté se přihlásíme se na konkrétní účet

```
USER užívatel
PASS heslo
```

a můžeme zadávat příkazy.

Ukázkový příklad:

```
$ telnet mail 110
Trying 10.16.66.18...
Connected to sunrise.
Escape character is '^]'.
+OK sunrise Cyrus POP3 v1.5.19 server ready
USER radek
+OK Name is a valid mailbox
PASS mojeheslo
+OK Maildrop locked and ready
STAT
+OK 3 109201
LIST
+OK scan listing follows
```

```
1 1207
2 6270
3 101724
.
QUIT
+OK
Connection closed by foreign host.
```

33.1.1.2. Přehled příkazů POP3 protokolu

USER *userid*

První příkaz který zadáváme po spojení se serverem. Provádím přihlášení ke svému účtu (poštovní přihrádce)

PASS *password*

Tento příkaz následuje ihned za příkazem **USER**. Zadané heslo musí odpovídat našemu heslu k poštovní schránce jejíž název jsme uvedli v příkazu **USER**.

STAT

Na tento příkaz nám server vrátí informace o zaplnění schránky. Odpověd má strukturu

```
+OK msgs bytes
```

Kde první část `msgs` je počet zpráv v přihládce a `bytes` je celkové zaplnění přihrádky v bytech.

```
STAT
+OK 3 109201
```

LIST

Na tento příkaz nám server vrátí seznam dopisů v přihrádce ukončený řádkem s tečkou. Na jednotlivých řádcích výpisu jsou vždy číslo pořadové číslo dopisu a jeho velikost v bytech. Ukázkový seznam:

```
LIST
+OK scan listing follows
1 1207
2 6270
3 101724
.
```

RETR *msg#*

Přečte zprávu s číslem *msg#*.

TOP *msg# #lines*

Vypíše hlavičku a prvních několik řádků zprávy. Parametr `msg#` určuje zprávu a parametr *#lines* pak počet řádků těla zprávy.

DELE *msg#*

Označí správu číslo `msg#` ke smazání. Ke skutečnému smazání/odstranění zprávy dojde až po zadání příkazu **QUIT**.

RSET

Zruší příznak mazaní u správ kde byl nastaven, takže je příkaz **QUIT** neodstraní.

QUIT

Smaže všechny zprávy označené ke smazání a odhlásí od serveru.

33.1.2. Internet Message Access Protocol (IMAP)

* *section id="imap" condition="author"*

Odkazy:

- protocole imap: log d'une session telnet (http://www.salemioche.com/imap/imap_session.php)
- How to verify basic IMAP connectivity by using Telnet (http://support.microsoft.com/kb/q189326/)
- Configuring Exim and Courier IMAP under Debian GNU/Linux (http://talk.trekweb.com/~jasonb/articles/exim_maildir_imap.shtml)
- Troubleshooting IMAP (http://www.macgeekery.com/node/26)

Protokol IMAP na rozdíl od portokolu POP3 počítá s tím že uživatelovi maily zůstávájí na serveru kde mohou být organizovány do složek. Je tedy vhodný pro případ kdy uživatel přistupuje ke své poště z více míst (strojů), právě proto že dopisy zůstávají na IMAP serveru.

IMAP server používá porty 143, 220 a 993 jak je vidět na následujícím výpise.

```
# grep -i imap /etc/services
imap2           143/tcp         imap            # Interim Mail Access Proto v2
imap2           143/udp         imap
imap3           220/tcp                         # Interactive Mail Access
imap3           220/udp                         # Protocol v3
imaps           993/tcp                         # IMAP over SSL
imaps           993/udp                         # IMAP over SSL
```

Na mail serveru mám nainstalován cyrus

```
# dpkg -l "cyrus*"
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version        Description
+++-==============-==============-============================================
ii  cyrus-admin    1.5.19-9.2     CMU Cyrus mail system (administration tool)
ii  cyrus-common   1.5.19-9.2     CMU Cyrus mail system (common files)
pn  cyrus-dev      <none>         (no description available)
ii  cyrus-imapd    1.5.19-9.2     CMU Cyrus mail system (IMAP support)
pn  cyrus-nntp     <none>         (no description available)
ii  cyrus-pop3d    1.5.19-9.2     CMU Cyrus mail system (POP3 support)
```

Základní interakce ze serverem vypadá následovně:

```
$ telnet mail 143
Trying 10.16.66.18...
Connected to sunrise.
Escape character is '^]'.
* OK sunrise Cyrus IMAP4 v1.5.19 server ready
0 CAPABILITY
* CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS X-NON-HIERARCHICAL-RENAME NO
0 OK Completed
0 LOGOUT
* BYE LOGOUT received
0 OK Completed
```

33.1.2.1. Příkazy

CAPABILITY

Na tento příkaz nás IMAP server informuje o svých možnostech.

```
* OK sunrise Cyrus IMAP4 v1.5.19 server ready
a CAPABILITY
* CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS X-NON-HIERARCHICAL-R
a OK Completed
```

LOGIN

Slouží k přihlášení.

```
* OK sunrise Cyrus IMAP4 v1.5.19 server ready
a LOGIN uzivatel heslo
a OK User logged in
```

LOGOUT

Odhlášení od serveru.

```
a LOGOUT
* BYE LOGOUT received
a OK Completed
Connection closed by foreign host.
```

LIST

.

SELECT

.

FETCH

.

Další příkazy jsou: STORE, EXPUNGE

33.1.3. SMTP

* *section id="smtp" condition="author"*

Odkazy:

- How do you test POSTFIX for sending mail out? (http://www.linuxquestions.org/questions/linux-software-2/how-do-you-test-postfix-for-sending-mail-out-106027/)

```
$ telnet <IP> 25
EHLO
MAIL FROM: <from-email>
RCPT TO: <recipient-email>
DATA
Type message here.
. <Enter>
```

```
$ telnet mail.firma.cz 25
Trying 10.16.66.18...
```

```
Connected to sunrise.
Escape character is '^]'.
220 sunrise.firma.cz ESMTP Postfix
HELO sunrise.firma.cz
250 sunrise.firma.cz
MAIL FROM:<root@firma.cz>
250 Ok
RCPT TO:<nospam@hnilica.cz>
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Halo
testuji test
-- Radek
.
250 Ok: queued as 092F8EC8D
quit
221 Bye
Connection closed by foreign host.
$
```

Tabulka 33-1. Kódy odpovědí SMTP

2xx	Příkaz byl úspěšně vykonán.
3xx	Příkaz byl přijat, server očekává další informace.
4xx	Nastala chyba. Dočasná chyba.
5xx	Nastala chaba. Příkaz byl neúspěšný. Trvalá chyba.

33.2. MTA

33.2.1. Postfix

* *section id="postfix"*

Postfix je poštovní program (MTA), náhrada sendmailu a konkurence Qmailu. Wietse Venema ho napsal s maximálním ohledem na bezpečnost, jako náhradu za sendmail s jehož bezpečností nebyl spokojen.

Spolehlivost

FIXME:

Bezpečnost

Postfix je z hlediska bezpečnosti rozdělen na moduly, samostatné programy řešící jednotlivé fáze spracování pošty. To jednak zjednodušuje návrh těchto programů z bezpečnostního hlediska. Rozdělení právy kdy každý program může běžet jen z právy která nezbytně pro svou funkci představuje. Snížení rizika. Případná chyby v jednom programu neovlivní programy ostatní.

Výkon

FIXME:

Flexibilita

FIXME:

„Kompatibilita se Sendmailem“

FIXME:

33.2.1.1. Přehled

Některé poštovní progrmy jako například Sendmail jsou velké monolitické aplikace které dělají vše. Tento přístup je ale největším problémem bezpečnosti. Sdílení dat v jedné aplikaci je sice jednodušší, ale veliká monolitická aplikace je noční můrou všech co se zabývají bezpečností.

Architektura Postfixu je založena na malých prográmcích jenž vzájemně kooperují.

Jádro postfixu je implementováno v několika „rezidentních“ programech. Tyto spolu komunikují přes unixové sokety, nebo roury.

Postfix má několik vnitřních front kterými data protékají: *maildorp*, *incoming*, *active* a *deferred*.

Dopisy posílané z místního stroje jsou ukládány do *maildrop* a po prvotním vyčištění jsou zkopírovány do *incoming*. Odtud jsou načítány a doručovány do cíle. Dopisy jenž nelze doručit končí ve frontě *deferred*.

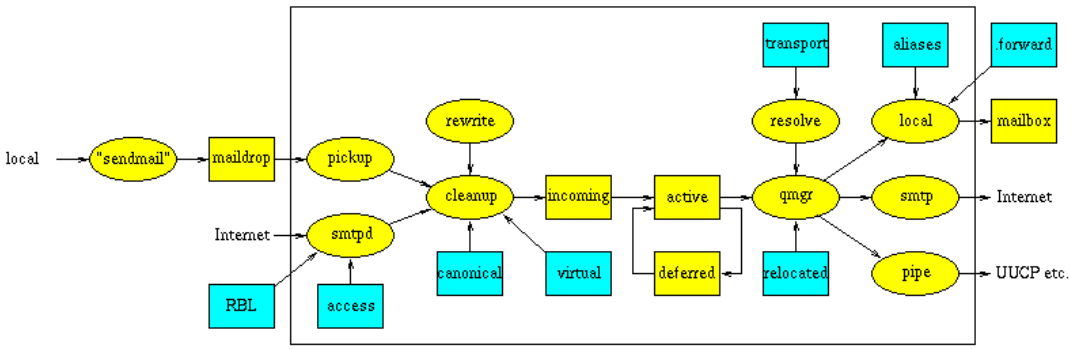
Manažer front si ještě udržuje frontu *active*. Tato je omezaná svou velikostí a je udržována v paměti. Při uvolnění místa jsou načítány dopisy střídavě z *incoming* a *deferred*. Tím je zajištěn pruchod nových dopisů i v případě velmi rozáhlé fronty *deferred*.

Mimo již zmíněné fronty používá postfix ještě dvě odkládací (parkovací): *hold* která drží zmražené maily a *corrupt* kam jsou odkládány poškozené dopisy.

33.2.1.2. Jak to funguje

Na začátek jeden obrázek který jsem převzal z webu PostFix.ORG (http://www.postfix.org/big-picture.html).

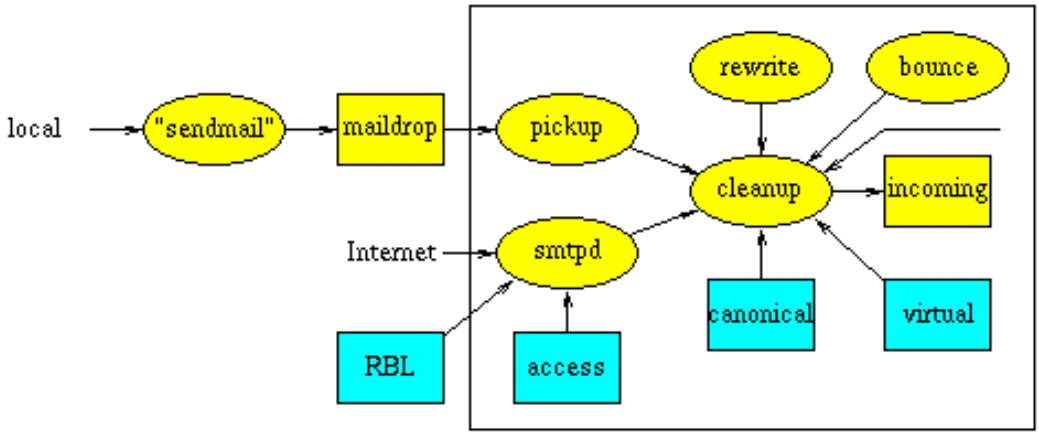
Obrázek 33-2. Graf toku dopisů postfixem



Na začátek jeden obrázek který jsem převzal z webu PostFix.ORG (http://www.postfix.org/receiving.html).

33.2.1.2.1. Příjem dopisů

Obrázek 33-3. Graf příjmu dopisů



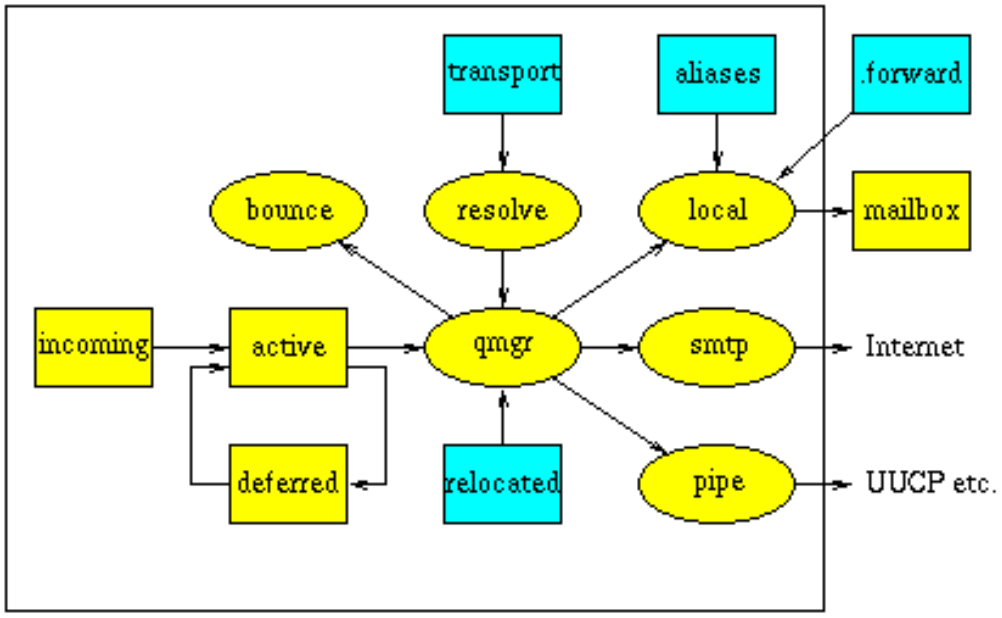
Na vstupu všechny dopisy padají do fronty *maildrop*

- Lokálně poslané dopisy, předané programem sendmail (komponentou Postfixu). sendmail spustí privilegovaný program postdrop, jenž uloží dopis do adresáře fronty *maildrop*. Odtud jsou vybírány démonem pickup. Démon provádí základní kontrolu aby ochránil zbytek systému Postfix.
- Dopisy které příjdou sítí. SMTP server postfixu přijme dopis, provede úvodní kontrolu aby ochránil zbatek systému.
- Dopisy generované samotným postfixem vytváří démoni bounce a defer
- O přeposílání (forwarding) dopisů se stará doručovací agent local. Využívá k tomu systémovou databázi `/etc/alias` a uživatelské soubory `$HOME/.forward`.
- Pošta generovaná postfixem, jenž upozorňuje ne problémy. Postfix může být zkonfigurován tak, aby upozorňoval správce pošty na problémy SMTP, porušení pravidel UCE atd.
- Démon `cleanup` provádí závěrečné zpracování dopisu. Přidává chybějící pole `From:` a další chybějící hlavičky, přepisuje adresu do standardní formy <`user@fully.qualified.domain`> a upozorňuje manažera fronty `queue manager` na příchod nové pošty. Démon `cleanup` může být konfigurován aby transformoval adresu na základě tabulek `canonical` a `virtual`.
- Na základě žádosti démona `cleanup` přepisuje démon `trivial-rewrite` adresu do standardní formy <`user@fully.qualified.domain`>. Postfix nmá žádný přepisovací jazyk kterým by se podobné transformace daly popsat. Místo toho používá vyhledávací tabulky (`table lookup`).

33.2.1.2.2. Doručování dopisů

Pošta jednou přijatá a umístněná do fronty `incoming` čeká na další krok, doručení. Následující obrázek převzatý z PostFix.ORG (http://www.postfix.org/delivering.html) ukazuje jak Postfix doručuje poštu.

Obrázek 33-4. Graf doručování dopisů



- Manažer fronty (*queue manager*) je srdcem Postfixu. Je v kontaktu s doručovacími agenty *local*, *smtp*, *lmtp* a *pipe*.

  Manažer fronty udržuje oddělenou frontu `deferred` pro poštu kterou nemůže z různých důvodů doručit.

  Manažer fronty undržuje malou frontu `active`. Tato fronta slouží pro aktuálne zpracovávané dopisy přebírané z front `incoming` a `deferred`.

  Manažer fronty může vracet dobisy (*bounce*) jejichž příjemci jsou uvedení v tabulce `relocated`. Tato tabulka obsahuje informaci o uživatelích, nebo celých doménách kteří již neexistují.

  Démon *trivial-rewrite* vyhodnocuje cílové adresy na žádost manažera fronty. Standardně rozlišuje mezi cílemi (destinacemi) *local* a *remote*. Případná další směrovací informace může být uvedena v tabulce `transport`.

  Démon *bounce or defer* generuje upozornění (report) na dopisy jenž nemohou být doručeny, z důvodů *unrecoverable* chyby, nebo proto, že cílový server nemohl být kontaktován ani v prodlouženém čase.

- Místní doručovací agent *local* umí doručovat poštu do UNIXových přihrádek (*mailbox*) a rozumí systémovým přezdívkám `/etc/alias` a uživatelským souborům `$HOME/.forward`. Najednou může být použito více doručovacích agentů, ale doručování jednomu uživateli je obvykle omezené.

33.2.1.3. Instalace

* *Vyhodit listingy, nebo je aspoň zkrátit.*

V Debian Woody máme v zásadě dvě skupiny balíčků. Jednu postavneou na `postfix 1.1.11-0.woody2` a druhou na `postfix-snap 0.0.20020115-5`

Instalace pomocí příkazu `wajig` je jednoduchá

```
# wajig install postfix
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  postfix-ldap postfix-pcre
The following packages will be REMOVED:
  exim
The following NEW packages will be installed:
  postfix postfix-ldap postfix-pcre
0 packages upgraded, 3 newly installed, 1 to remove and 0  not upgraded.
Need to get 558kB of archives. After unpacking 343kB will be used.
Do you want to continue? [Y/n] Y
```

Konfigurace

```
General type of configuration? Internet with Smarthost
Mail name? firma.cz
Append .domain to simple addresses? No
SMTP relay host? (blank for none) ketu.firma.cz
Other destinations to accept mail for? (blank for none) moon.firma.cz, localhost.firma.cz, loc

Get:1 http://debian woody/main postfix-ldap 1.1.11-0.woody2 [26.8kB]
Get:2 http://debian woody/main postfix-pcre 1.1.11-0.woody2 [23.5kB]
Get:3 http://debian woody/main postfix 1.1.11-0.woody2 [508kB]
Fetched 558kB in 1m1s (9115B/s)
Preconfiguring packages ...
dpkg: exim: dependency problems, but removing anyway as you request:
 mutt depends on exim | mail-transport-agent; however:
  Package exim is to be removed.
  Package mail-transport-agent is not installed.
  Package exim which provides mail-transport-agent is to be removed.
 mutt depends on exim | mail-transport-agent; however:
  Package exim is to be removed.
  Package mail-transport-agent is not installed.
  Package exim which provides mail-transport-agent is to be removed.
 at depends on mail-transport-agent; however:
  Package mail-transport-agent is not installed.
  Package exim which provides mail-transport-agent is to be removed.
 mailx depends on mail-transport-agent; however:
  Package mail-transport-agent is not installed.
  Package exim which provides mail-transport-agent is to be removed.
(Reading database ... 15808 files and directories currently installed.)
Removing exim ...
Selecting previously deselected package postfix-ldap.
(Reading database ... 15746 files and directories currently installed.)
Unpacking postfix-ldap (from .../postfix-ldap_1.1.11-0.woody2_i386.deb) ...
Selecting previously deselected package postfix-pcre.
Unpacking postfix-pcre (from .../postfix-pcre_1.1.11-0.woody2_i386.deb) ...
Selecting previously deselected package postfix.
Unpacking postfix (from .../postfix_1.1.11-0.woody2_i386.deb) ...
Adding ‘diversion of /usr/share/man/man8/smtpd.8.gz to /usr/share/man/man8/smtpd.real.8.gz by
Setting up postfix (1.1.11-0.woody2) ...
Adding group postfix (101)...
Done.
adduser: Warning: The home dir you specified already exists.
Adding system user postfix...
Adding new user postfix (102) with group postfix.
Not creating home directory.
```

```
Adding group postdrop (102)...
Done.
setting myhostname: moon.firma.cz
setting alias maps
setting alias database
changing /etc/mailname
setting myorigin
setting destinations: moon.firma.cz, localhost.firma.cz, localhost
setting append_dot_mydomain: no
setting relayhost: ketu.firma.cz
setting mynetworks: 127.0.0.0/8
setting mailbox_command
setting mailbox_size_limit: 0
setting recipient_delimiter: +
/etc/aliases does not exist, creating it.

Postfix is now set up with a default configuration.  If you need to make
changes, edit
/etc/postfix/main.cf (and others) as needed.  To view Postfix configuration
values, see postconf(8).

After modifying main.cf, be sure to run '/etc/init.d/postfix reload'.

Running newaliases
Starting mail transport agent: Postfix.

Setting up postfix-ldap (1.1.11-0.woody2) ...

Setting up postfix-pcre (1.1.11-0.woody2) ...

moon:~#
```

33.2.1.4. Konfigurace

Soubor `main.cf`

```
setgid_group = postdrop
mydestination = localhost, $myhostname, firma.cz, jina-firma.cz, pokus.cz
relay_domains = $mydestination, hash:/etc/postfix/maps/relay-domains

mailbox_command = procmail -a "$EXTENSION"

# Antispam
local_recipient_maps = $alias_maps unix:passwd.byname
disable_vrfy_command = yes
allow_untrusted_routing = no

smtpd_helo_required = yes
smtpd_helo_restrictions =
    permit_mynetworks,
    reject_invalid_hostname,
    reject_unknown_hostname
    reject_non_fqdn_hostname


smtpd_client_restrictions =
```

```
        permit_mynetworks,
        permit_sasl_authenticated,
        check_recipient_access hash:/etc/postfic/maps/norbl,
        reject_rbl_client blackholes.easynet.nl,
        reject_rbl_client dnsbl.ahbl.org,
        reject_rbl_client dnsbl.njabl.org,
        reject_rbl_client list.dnsbl.org,
        ⋮
        reject_rbl_client multihop.dnsbl.org,
        reject_rbl_client dynablock.easynet.nl,
        reject_unknown_client

    smtpd_delay_reject = yes

    smtpd_sender_restrictions =
        reject_unknown_sender_domain,
        reject_non_fqdn_sender,
        ...

    strict_rfc821_envelopes = yes


    ### SASL suppoert
    smtpd_sasl_auth_enable = yes
    smtpd_sasl_security_options = noanonymous
    smtpd_sasl_local_domain = $myhostname
    broken_sasl_auth_clients = yes

    ### TLS
    smtpd_use_tls = yes
    smtpd_tls_auth_only = yes
    smtpd_tls_cert_file = /etc/CA/mail.firma.cz-server-cert.pem
    smtpd_tls_key_file = /etc/CA/mail.firma.cz-server.key.pem
    smtpd_starttls_timeout = 300s
```

`setgid_group`

FIXME:

`mydestination`

FIXME:

`smtpd.conf`:

mech_list: PLAIN LOGIN
pwcheck_method: saslauth

```
# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 sun.firma.cz ESMTP Postfix
EHLO localhost
250-sun.firma.cz
250-PIPELINING
250-SIZE 50000000
250-VRFY
```

```
250-ETRN
250-XVERP
250 8BITMIME
QUIT
221 Bye
Connection closed by foreign host.
```

* ***FIXME:****prozkoumat: `awstats`. Jedná se o obecný analyzér. S Postfixem se moc nekamarádí, je nutno překonvertovat vstupní data.*
* ***FIXME:****SARG (http://sarg.sf.net), analyzér ke squidu. Nepočítá z Postfixem.*

33.2.1.5. Jak přimět postfix k přeposílání pošty vybraných uživatelů jinam

Zkusím použít *transport_maps*

relocated_maps

33.2.1.6. Mapování virtuálních adres

Virtual address mapping

Mapování virtuálních adres u přijaté pošty při doručení se řídí podle *alias_maps*. Zde nastavíme například

```
alias_maps = hash:/etc/aliases, ldap:ldapsource
ldapsource_server_host = ldap.firma.cz
ldapsource_search_base = dc=firma,dc=cz
ldapsource_query_filter = (&(mail=%s)(!(|(maildrop="*|*")(maildrop="*:*"))))
```

`ldapsource` je jen symbolické jméno, nikoliv klíčové slovo. Nahradím-li jej více popisným názvem, v tom smyslu, že použiji LDAP na více místech a potřebuji je od sebe rozlišit, bude to vypadat například takto

```
alias_maps = hash:/etc/aliases, ldap:ldapalias
ldapalias_server_host = ldap.firma.cz
ldapalias_search_base = dc=firma,dc=cz
ldapalias_query_filter = (&(mail=%s)(!(|(maildrop="*|*")(maildrop="*:*"))))
```

33.2.1.7. Jak protékají dopisy Postfixem

Postfix má dva vstupní body

local
internet

.

Vstupní bod *local*.

Vstupním bodem *Internet*. proudí do serveru pošta ze sítě přes SMTP protokol.

33.2.1.8. Transport

Transport provádí směrování pošty pro celé domény. Použijeme jej například při *mail relayingu* na hradebním počítači (*firewall*).

Aby nám transport chodil, musíme zapsat do souboru `/etc/postfix/main.cf` řádek:

```
transport_maps = hash:/etc/postfix/transport
```

v souboru `/etc/postfix/transport` pak popisujeme jednotlivé transporty. Řádky mají tvar

```
domain transport:nexthop
```

nebo

```
.domain transport:nexthop
```

v druhém případě se transport aplikuje pro všechny subdomény.

Doména *`domain`* je doménou na niž se transport uplatní, *`transport`* je konkrétní transport, například `smtp`, `uucp`, ... Poslední část *`nexthop`* je parametr transportu. Pro každý transport je jiný.

Konkrétní příklad přesměrování pošty pro doménu `sunrise.firma.cz` přes počítač `misuro.firma.cz`

```
sunrise.firma.cz smtp:misuro.firma.cz:25
```

33.2.1.9. Poznámky, řešení, příklady

Zatím nezapracované poznámky k PostFixu

```
# postmap /etc/postfix/virtual
# postfix reload
# postconf mynetworks
```

33.2.1.9.1. Kopie veškeré průchozí pošty

ToDo:

1. Prozkoumat prametr „always_bcc“

FIXME:

33.2.1.9.2. Nastavení přímého doručování v MAC OS-X 10.3

FIXME:

Aby běžel smtp démon Postfix trvale, je třeba v `/etc/postfix/master.cf` odkomentovat řádek

```
smtp      inet  n       -       n       -       -       smtpd
```

V souboru `/etc/hostconfig` nastavit `MAILSERVER=-YES-` a restartovat.

```
# postfix start
```

Poté se můžeme přesvědčit jestli lokální smtp běží a poslouchá na portu:

```
# netstat -p smtp
```

Používanému poštovnímu klientu pak nastavímé náš, právě spuštěný, lokální, smtp server.

33.2.1.9.3. Filtrování pošty

Pořebujeme li filtrovat poštu, teď mám na mysli odmítnout dopisy podle hlavičky (obsahem si nejsem jist), můžeme použít konfigurační soubor `/etc/postfix/reject`. Do tohoto souboru napíšeme regulární výraz společně s příkazem, například takovýmto:

```
/^Subject: [A-Z ]{4,}/ REJECT
```

Tento konkrétní řádek působí domítnutí pošty jejíž subjekt začíná více než čtyřmi velikými písmeny a mezerami. Do souboru `/etc/postfix/main.cf` ještě přidáme řádek

```
header_checks = regexp:/etc/postfix/reject
```

33.2.1.9.4. Použití LDAPu

Na spoustě míst používá Postfix vyhledávací tabulky k nějakému rozhodování. Například pro vyhodnovení virtuální adresy na skutečnou. Jendím z mechanismů které pro to můžeme použít je LDAP databáze. V místě kde přířazujeme tabulky například virtuálních adres

```
virtual_maps = hash:/etc/postfix/virtual, ldap:ldapvirtual
```

Na tomto řádku je vidět, že jako jeden zdroj je použit ldap. Klíčové slovo `ldap:`je následováno jménem které jsme si zvolili. Toto jméno pak slouží jako prefix u všech parametrů.

`server_host`

```
ldapvirtual_server_host = ldap.firma.cz
```

Server na kterém běží LDAP server. V každém LDAP zdroji který použijeme se můžeme odkazovat na jiný server.

`search_base`

```
ldapvirtual_search_base = dc=firma,dc=cz
```

Bázová adresa pro dotazy. LDAP podstrom který budeme prohledávat.

`bind`

```
ldapvirtual_bind = no
```

Příznak zdali se na server vážeme/připojujeme pod vyhrazeným účtem.

`query_filter`

```
ldapvirtual_query_filter = (mailLocalAddress=%s)
```

Vlastní text dotazu na LDAP server.

`result_attribute`

```
ldapvirtual_result_attribute = mailRoutingAddress
```

Název atributu záznamu obsah kterého bude vrácen jako odpověď dotazu.

33.2.1.9.5. Omezení mailů na příjmu

FIXME: Na emaily se dají klást omezení jinž při příjmu. Tato omezení spadají do několika kategorií. Dle velikosti, dle chybovosti klienta, ...

Omezení podle velikosti mailu provedemen nsatavením proměnné `message_size_limit` v hlavním konfiguračním souboru `main.cf`. Standardní omezení je 10MB. Chceme li tento limit snížit například na 2MB bude příkaz vypadat takto:

```
message_size_limit = 2097152
```

Umezení se uplatňuje na velikost emailu jako takového. Je třeba si uvědomit že přílohy bývají kódovány a jako takové jsou v zakódovaném tvaru až o 30% větší.

Dalším omezením je množství příjemců u mailů rozesílaných na více adres. Direktiva `smtpd_recipient_limit` jenž má defaultně hodnotu 1000 omezuje množství příjemnců u jednoho mailu.

Dalším způsobem omezení na vstupu je omezení dle chybovosti klienta. Toto se nastavuje direktivami `smtpd_error_sleep_time`, `smtpd_soft_error_limit` a `smtpd_hard_error_limit`.

33.2.1.9.6. Nezapracováno

FIXME:

```
### Parrallel delivery force (local=2 and dest=20 are aggressive)
local_destination_concurrency_limit = 20
default_destination_concurrency_limit = 20

### Max flow rate (1 sec delay per 50 emails/sec over the number of emails delivered/sec)
in_flow_delay = 1s

### Parrallel delivery force (local=2 and dest=20 are default)
local_destination_concurrency_limit = 2
initial_destination_concurrency = 10
default_destination_concurrency_limit = 50

### Limits the mail inflow to 100 messages per second above the number of messages delivered p
in_flow_delay = 1s
```

33.2.2. QMail

* *section id="qmail"*

Poštovní server psaný s ohledem na maximální bezpečnost.

33.2.2.1. Přeposílání pošty *Mail Relay*

Potřebujeme zprovoznit přeposílání pošty na hradebním počítači na poštovní server uvnitř firmy.

1. Do souboru `control/smtproutes` napíšeme jeden řádek, přesmerování veškeré pošty na vnitřní poštovní server. Například má-li náš poštovní server adresu `10.16.66.18` napíšeme:

   ```
   :10.16.66.18
   ```

2. Do souboru `control/rcpthosts` napíšeme pro které domény náš server přijímá poštu. Například

```
firma.cz
.firma.cz
```

33.2.3. Exim

Poštovní server standardně instalovaný v distribuci Debian/GNU Linux Potato a Woody.

33.2.4. sendmail

* *section id="sendmail" condition="author"*
* ***FIXME:****Dopsat.*

> *Administrátor který nikdy nekonfiguroval sendmail není administrátor. Administrátor který tak učinil podruhé je blázen.*
>
> *původ neznámý*

Sendmail je původní poštovní server. Vznikl v začátcích internetu. Jedná se o monolitický program který není považován za příliš bezpečný s velmi komplikovanou konfigurací. Velmi často se pro vytváření konfiguračních souborů sendmailu používa preprocessor **m4 m4**.

.

`/etc/access`

FIXME:

```
# makemap hash /etc/mail/access.db < /etc/mail/access
```

`/etc/local-host-names`
`/etc/sendmail.cw`

localhost
nmnm.cz

`/etc/aliases`

postmaster: root
root: admin
⋮

```
# newaliases
```

`/etc/sendmail.mc`

FIXME:

```
include('/usr/share/sendmail/cf/m4/cd.m4')dnl
OSTYPE('debian')dnl

FEATURE('always_add_domain')dnl virtuální domény
FEATURE('use_cw_file')dnl
FEATURE('use_ct_file')dnl       statistiky
FEATURE('nouucp', 'reject')dnl  odmítat uucp
FEATURE('mailertable')dnl       tabulka pro přepisování a směrování domén
```

```
FEATURE('smsrsh')dnl            restricted shell

MAILER_DEFINITIONS
MAILER(local)dnl
MAILER(smtp)dnl

LOCAL_CONFIG
```

`/etc/relay-domains`

```
# vsechny domeny, pro ktere vas server zalohuje pres MX
mesto-domazlice.cz
mks.mesto-domazlice.cz
⋮
gym.nmnm.cz
```

vacation. Do souboru `~/.forward`

```
\snek, |"/usr/bin/vacation snek"
```

33.2.5. Jednoduchý SMTP Daemon (ssmtpd)

* *section id="ssmtpd" xreflabel="ssmtpd"*

Ve zkratce popíši instalaci a použití ssmtpd démona. (*Simple SMTP Daemon*)

SSMTP je velmi jednoduchý smtp démon který slouží jen pro odesílání pošty. Je napsán pro nasazení na strojích na které pošta nechodí, ale odkud je třeba ji odesílat. Například routery a specializované servery. Zde často vznikají maily jako výsledek pravidelných kontrol systému, či hlášení vyjímečných stavů. Přitom na tyto stroje pošta doručována není.

Instalace je jednoduchá

```
# apt-get install ssmtp
```

Při instalování nám pokládá řadu otázek a podle odpovědí se nakonfiguruje.

* *Popsat otázky při instalaci.*

Při instalace dojde k odstranění prvotní instalace programu exim.

Konfigurace je uložena ve dvou souborech: `/etc/ssmtp/ssmtp.conf` a `/etc/ssmtp/revaliases`. V prvním je vlastní konfigurace a druhý slouží ke směrování pošty. Je to taková náhrada za `/etc/aliases` ale pro odchozí poštu.

Ukázkové konfigurační soubory. Přepsal jsem jen soubor `ssmtp.conf`:

```
root=administrátor@firma.cz
mailhub=mail.firma.cz
rewriteDomain=tento_host.firma.cz
hostname=tento_host.firma.cz
FromLineOverride=YES
```

Kde *`firma`* je doménové jméno naší firmy, *`administrátor`* je poštovní adresa administrátora v naší firmě a *`tento_host`* je název počítače (hosta) na kterém ssmtpd instalujeme. Druhý soubor `revaliases` jsem nechal prázdný.

33.3. MDA WORKING

* *section id="mda" condition="author"*

FIXME:šablona

33.3.1. Cyrus (cyrus-imapd, cyrus-pop3d)

* *section id="cyrus-imapd"*

Odkazy:

- Cyrus Notes (http://www.wlug.org.nz/CyrusNotes) on WLUG Wiki
- Secure Email Using Cyrus IMAP, Sendmail, and SASLv2 (http://www.doorbot.com/guides/sendmail/securemail/)
- Postfix+Cyrus Imap+Sasl+tls (http://cernicalo.escomposlinux.org/~emeteo/imap/imap+postfix/) by Mario Teijeiro Otero

Při instalaci se dovyberou balíčky:

cyrus-common
cyrus-admin
tcl8.0

Nainstaluje se jediný konfigurační soubor `/etc/imapd.conf`. Bylo třeba tento upravit takto:

Úpravy konfiguračního souboru `/etc/imapd.conf`

1. Přidat seznam oprávněných administrátorů

```
admins: radek
```

33.3.1.1. PAM

Přesměroval jsem autentifikaci na PAM

Přesměrování autentifikace na PAM

1. Vyměnil jsem souboru `/etc/pam.d/cyrus` za

```
auth        required    pam_ldap.so
account     required    pam_ldap.so
password    required    pam_ldap.so
```

33.3.1.2. Postupy

Vytvoření nové poštovní přihrádky.

```
$ cyradm -user radek mail-pha
> createmailbox user.marie
> setquota user.marie 5000
> exit
```

Prohlížení přihrádek,. tedy jejich existence a nastavení, nikoliv obsahu. Předpokládám že jsem již přihlášen pomocí cyradm jako uživatel který má právo spravovat poštovní přihrádky.

```
> listmailbox user.*
```

33.3.1.2.1. Změna kvóty

Právě jsem byl informován, že uživatel <user> má problém s poštou. Podle hlášení poštovního programu postfix se jedná o překročení kvóty poštovní přihrádky.

Nejdříve se tedy podíváme, jak to vypadá

```
$ cyradm -user radek mail-hub
> lq user.user
STORAGE 20377/20000 (101%)
```

Tedy jak je vidět, uživatel má v poštovní přihrádce cca 20MB. Zvětšíme mu tedy kvótu na 100MB

```
> sq user.user 100000
```

ještě se podíváme jak schránka aktuálně vypadá

```
> lq user.user
STORAGE 14310/100000 (14%)
```

33.3.1.2.2. Přenášení doručených mailů mezi schránkami

Někdy se vyskytne potřeba přehodit maily ze schránky jednoho uživatele do schránky druhého. Například když pracovník nedojde do práce, nebo je nemocen a není předem nastaveno přesměrování pošty. V tomto případě prostě zkopírujeme vybrané nebo všechny dopisy ze schránky jednoho uživatele do schránky jiného uživatele. Před samotným kopírováním zastavíme programy které by nám mohly zasáhnout do dat.

```
# /etc/init.d/postfix stop
# /etc/init.d/inetd stop
```

Poté provedeme vlastní přenesení mailů.

```
# cd /var/spool/cyrus/mail/user
# cp tomas/1234. franta/
```

Poté provedeme opravu indexových souborů cílové přihrádky.

```
# su - cyrus
$ /usr/sbin/reconstruct user.franta
```

V případě že maily nekopírujeme ale přenášíme (**mv**) je potřeba opravit ještě přihrádku odkud maily přenášíme. Když už máme včechno hotovo, opět spustíme zastavené služby.

```
# /etc/init.d/inetd start
# /etc/init.d/postfix start
```

Uvedený postup je platný jen pro postfix spouštěný jako *standalone* a cyrus jehož pop3 a imap servery se spouští přes inetd.

33.3.1.3. Pokus o překlad cyrus21 pod Woody

> **Varování**
>
> Toto je opravdu jen pokus.

Upravím si konfigurační soubor `/etc/apt/sources.list` tak aby byly dosažitelné všechny potřebné balíčky. Musíme v něm mít řádek odkazující se na balíček `debhelper` z Debian Backports (Backports.ORG) (http://www.backports.org/), a řádek odkazující se na zdroje ze Sarge.

```
deb http://localhost:9999/backports woody debhelper
deb-src http://localhost:9999/main sarge main
```

K překladu budeme potřebovat nejdříve balíček `libsasl2-dev` který se rovněž ve Woody ne nachází.

```
# apt-get install zlib1g-dev

# apt-get source -t sarge -b libsasl2
⋮
dpkg-checkbuilddeps: Unmet build dependencies: libdb4.2-dev (>= 3.2.9-14), libopie-dev (>= 2.3
⋮

# apt-get install -t backports debhelper
# apt-get install tcl8.3-dev libdb3-dev libwrap0-dev libpam0g-dev libssl-dev
# apt-get install libzephyr-dev comerr-dev drac-dev libsnmp4.2-dev ...
# apt-get install xutils flex bison autotools-dev transfig gs groff
```

dpkg-checkbuilddeps: Unmet build dependencies: libsasl2-dev (>= 2.1.9)

33.3.1.4. Instalace cyrus 2.1.17

Jedná se o verzi 2.1.17 backportovanou do Debian Woody. Balíčky a potřebné informace jsem získal ze stránky Henrique de Moraes Holschuh Debian Collectibles (http://people.debian.org/~hmh/). Použité zdroje jsou:

```
deb http://www.backports.org/debian woody openldap2 openssl cyrus-sasl2 uw-imap  adduser
deb http://www.backports.org/debian woody ucf po-debconf automake1.8 autoconf debhelper
deb http://people.debian.org/~hmh/woody/ hmh/cyrus/
```

FIXME: Domnívám se že řada balíčků není potřebná, ale neměl jsem chuť a čas to řešit.

```
# apt-get install cyrus21-imapd
```

33.3.1.5. Překlad Cyrus 2.1 s podporou Toltec Connectoru

Nasazování Toltec Connectoru se neobešlo bez problémů, ty se podařilo postupně překonat naszaním Cyrus verze 2.1. Ovšem jeden problém zbyl. Toltec Connector si nemohl založit adresáře. Nezbylo než se podívat do zdrojů a pokusit se aplikovat patch od Toltec Connectoru.

Před vlastním Cyrusem si musíme připravit několik knihoven. Použijeme zdroje:

```
# Cyrus Imapd for Toltec Connector
deb http://localhost:9999/backports woody openldap2 openssl cyrus-sasl2 uw-imap  adduser
deb http://people.debian.org/~hmh/woody/ hmh/cyrus/
#deb http://localhost:9999/backports woody ucf po-debconf automake1.8 autoconf debhelper
```

```
deb-src http://localhost:9999/backports woody openldap2 openssl cyrus-sasl2 uw-imap  adduser
deb-src http://localhost:9999/backports woody ucf po-debconf automake1.8 autoconf debhelper
deb-src http://people.debian.org/~hmh/woody/ hmh/cyrus/

# apt-get install libsasl2-dev drac-dev
# apt-get source -b cyrus21-imapd
```

Tak výše uvedeným postupem jsem si ověřil, že mám vše potřebné a Cyrus přeložit jde. zůstává tedy aplikovat patch a přeložit Cyrus ručně.

FIXME:

33.4. MUA WORKING

* *section id="mua" condition="author"*

FIXME:šablona

33.4.1. mailx

FIXME:šablona

33.4.2. mutt

FIXME:šablona

33.4.3. mutt

33.4.3.1. Čeština

Nastavení kódování češtiny v odchozích správách pro všechny uživatele provedeme dopsáním řádku

```
set send_charset="us-ascii:iso-8859-1:iso-8859-2:utf-8"
```

do souboru `/etc/Murrrc`

33.4.3.2. Message scoring

Bodování zpráv (*message scoring*) je důležitým momentem pro třídění. Zprávě můžeme podle různých kritérii zvýšit bodové ohodnocení a tak zajistit že bude zatříděna výše než ostatní.

```
score ~U 10
```

33.5. courier-imap

* *section condition="author"*
* ***FIXME:****vyzkoušet, doplnit, konfigurace ve vztahu s exim.*

FIXME:

33.6. ipop3d

* ***FIXME:****doplnit*

.

33.7. fetchmail

```
$ fetchmail -a --smtpname radek@localhost natrix.cz
```

33.8. WebMail

- squirrelmail
- imp z horde
- nooc
- openwebmail
- twig
- aeromail

33.8.1. roundcube

Odkazy:
- roundcube (http://roundcube.net/)
-

```
$ aptitude search roundcube
p   roundcube                                                 - skinnable AJAX based web
p   roundcube-core                                            - skinnable AJAX based web
p   roundcube-mysql                                           - metapackage providing My
p   roundcube-pgsql                                           - metapackage providing Po
p   roundcube-sqlite                                          - metapackage providing sq
```

33.9. POP3 a IMAP servery

Programy sloužící jako poštovní servery přístupné protokolem `pop3`

`popa3d`

Malý pop3 démon napsaný s ohledem na bezpečnost.

`qpopper`

Rozšířený server POP3.

`courier-pop`
`courier-pop-ssl`
`courier-imap`
`courier-imap-ssl`

POP3 démon s SSL, PAM a podporou pro Maildir.

`ipopd`
`ipopd-ssl`

POP2 a POP3 server z UW.

`uw-imapd`
`uw-imapd-ssl`

IMAP démon z UW.

`teapop` — *stable teapop 0.3.4-1woody2 (54.5k)*
`teapop-pgsql`
`teapop-mysql`

Flexibilní a mocný POP3 server odpovídající RFC.

Teapop is a POP-3 server (compliant with RFC1939 and RFC2449) which supports:

- Virtual hosting ("VPOP")
- Flexible authentication (can get username/password from mysql, PostgreSQL, .htpasswd files, system password db... LDAP coming RSN)
- APOP
- mbox and Maildir-style spools
- Use of X-UIDL headers
- Ignoring UW-IMAPD control mails
- Running from inetd or standalone
- Various possible POP-before-SMTP methods if you know what you’re doing.

`cyrus-pop3d`
`cyrus-imapd`

CMU Cyrus mail system (POP3 support, IMAP support).

Cyrus is a fully-featured IMAP daemon, with a number of features not found in other IMAP implementations, including:

- Designed to handle massive quantities of mail
- No need for users to have login accounts
- Support for POP3 in addition to IMAP
- Servers don’t run as root

- Easy support for mail quotas

For more information, see http://asg.web.cmu.edu/cyrus/.

Poznámka: Note: Cyrus doesn't support reading from and storing mail in your standard mail spool - it stores mail in a separate directory in its own MH-like format.

`mailutils-pop3d` — `stable mailutils-pop3d 20020409-1 (26.5k)`

GNU Mailutils-based POP3 démon.

`solid-pop3d`

POP3 server s podporou Maildir, PAM a virtuálního hostingu.

The Solid POP3 Server is an implementation of a Post Office Protocol version 3 server that has flexibility as its main goal. The server is easily configurable and has support for features such as:

- APOP authentication scheme
- virtual hosting
- maildir and mailbox handling
- bulletins
- expiration of messages

`cucipop` — `stable cucipop 1.31-15.8 (36.9k)`

POP3 démon od Cubic Circle.

Poznámka: Non-Free

33.10. Antivirová kontrola dopisů

Zajímavou možností je na centrálním poštovním uzlu provádět antivirovou kontrolu dopisů

Na Linux-Sec.net (http://www.linux-sec.net) je přehled antivirů (http://www.linux-sec.net/Mail/antivirus.gwif.html).

33.10.1. Amavis-Postfix

Amavis-postfix je baliček programu amavis do poštovního programu Postfix. Umožňuje připojení konkrétního antivirového programu, například ClamAV.

33.11. Různé poznámky

33.11.1. Sprovoznění virtuálních poštovních domén

Pro tento účel je možno použít kombinaci programů Qmail a vchkpw

```
Apt-get install qmail-src ucspi-tcp-src vchkpw
Build-qmail
Build ucspi-tcp
```

S programem qmail je možno použít vlastní skript pro kontrolu hesla `checkpasswd`.

Další možností je pro POP/IMAP použít courier-pop a courier-imap

V sendmailu je to možno udělat takto:

```
for virtual domains pop/imap servers...

/etc/mail/local-host-names ( sendmail.cw )
        domain_1.com
        pop.domain_1.com
        mail.domain_1.com

        domain_two.com
        pop.domain_two.com
        mail.domain-two.com

        # this is the "real" machine name
        primary.com
        pop.primary.com
        mail.primary.com

pinging those domain names should have the same ip# if oyu wnat
to use just one server... ( fix your dns till it works right )

/etc/mail/virtusertable
        webmaster@domain_1.com  webmaster,webmaster_1@yahoo.com
        webmaster@domain_two.com  webmaster,webmaster_two@excite.com
        webmaster@primary.com   webmaster

who can send email thru your server
 ( arriving at the recepeint as coming from somebody@domain_1.com

        /etc/mail/relay-domains
        /etc/mail/relay_allow

if you wanna stop some spam... ( the hard way )
        /etc/mail/access
-->>
-->> make the new db in /etc/mail && dont forget to restart sendmail
-->>

now setting up pop/imap.... ( use secure pop3s or imaps instead )
        ..
        standard issue...
        ...
        /etc/hosts.allow
```

```
        /etc/hosts.deny
        ..

test it
        telnet pop.domain_1.com 110  ( regular pop )
        telnet pop.domain_1.com 995  ( might fail - checks protocol )

        - use a SSL enable client to do secure pop3/secure imaps
                IE, netscape, eudora, etc..etc..
                stunnel, ssh, etc...


Secure pop3 ( howto info )

        http://www.Linux-Sec.net/Mail/secure_pop3.txt
```

33.12. Spam a boj proti němu

V případě že se domníváme že spammer je z české republiky, můžeme podat stížnost na Úřad pro ochranu osobních údajů (http://www.uoou.cz). Konkrétně Úřad k zákonu č. 480/2004 (http://www.uoou.cz/spam_urad.php3) kde v dolní části stránky je tlačítko Podat stížnost (http://www.uoou.cz/int/spam_stiznost.php3).

33.12.1. Spam Assasin

FIXME:

Konfigurace samotného spamAssasina je v několika souborech. Jednak konfigurační soubory pro celý host v adresáři `/etc/spamassasin/`. Zde je konfigurace SpamAssasina pro celý host, t.j. společná, a pak uživatelská konfigurace v adresáři `~/.spamassasin/`. V konfiguračním soboru `/etc/spamassassin/local.cf` mám uvedeno:

```
report_safe 0
spam_level_stars 1

## Bayes filter configuration
use_bayes       1
use_bayes_rules 1
bayes_auto_learn 0
bayes_learn_during_report 1
bayes_auto_expire 1
```

Na těchto pravidlech je zajímavé nastavení `report_safe 0`, které říká že informace o důvodu proč byl mail klasifikován jako spam se uvádějí do hlavičky mailu a nikoliv do těla. Vkládání do těl způsobovalo že označené maily nebyly z důvodů „poškození“ struktury/kódování mailu čitelné.

> **Poznámka:** Bayes filter začne fungovat až poté, co má v databázi alespoň 200 vzorků spamů a 200 vzorků hamů. Je možné tyto hodnoty změnit, ale pokud jsem je měnil směrem dolů, nepomohlo to. Musel jsem počkat až budu mít minimálně 200 vzorků od každého.

33.13. Služby přístupné pomocí elektronické pošty

<dice@pbm.com>

Hrací kostka

33.13.1. PBM Hry

FIXME:

- GLAXY Explorer (http://explorer.sourceforge.net/)
- TNF2 (http://www.geocities.com/flandar/tnf2/index.html) — The New Frontiers 2

33.14. Webová rozhraní pro správu pošty

Odkazy:

- Postfix Mail Server Web interface,Frontend or GUI Tools (http://www.debianadmin.com/postfix-mail-server-web-interfacefrontend-or-gui-tools.html) [2006-10-31]
-

Software k prozkoumání:

- Replex (http://sourceforge.net/projects/replex/) — Administrativní nástroj pro Unixu podobné OS. Umožňuje spravovat více domén, distribuční seznamy. Užiatelé pošty jsou oddělení od uživatelů systému. Založeno na Postfixu, Cyrusu, PHP a MySql
- MyPFXAdmin (http://sourceforge.net/projects/mypfxadmin/) — MyPFXAdmin is a set of web based PHP scripts that allow easy administration of Postfix setup using the Postfix+Courier-IMAP+MySQL Multiple Domain HOWTO. It allows adding and editing of domains, aliases, and users. Poslední verze 0.8.1 dne 2003-10-09.
- PhpMailAdmin (http://freshmeat.net/projects/phpmailadmin/)
-

Debianí balíčky k prozkoumání (Lenny):

- courier-webadmin () —
-
-
-
-

33.14.1. courier-webadmin

Instalace:

```
wimp:~# aptitude search courier-webadmin
p   courier-webadmin                - Emailový server Courier - webový administr
wimp:~# aptitude install courier-webadmin
Čtu seznamy balíků... Hotovo
Vytvářím strom závislostí
Čtu stavové informace... Hotovo
```

```
Čtu rozšířené stavové informace
Inicializuji stavy balíků... Hotovo
Načítám popisy úloh... Hotovo
Následující NOVÉ balíky budou instalovány:
  courier-authdaemon{a} courier-authlib{a} courier-authlib-userdb{a}
  courier-base{a} courier-webadmin expect{a} fam{a} libfam0{a} libltdl3{a}
  portmap{a} tcl8.4{a}
0 balíků aktualizováno, 11 nově instalováno, 0 k odstranění a 0 neaktualizováno.
Potřebuji stáhnout 2183kB archivů. Po rozbalení bude použito 5759kB.
Chcete pokračovat? [Y/n/?]
```

```
cp /usr/lib/courier/courier/webmail/webadmin /usr/lib/cgi-bin/courierwebadmin
chmod u+s /usr/lib/cgi-bin/courierwebadmin
```

http://localhost/cgi-bin/courierwebadmin

```
# touch /etc/courier/webadmin/unsecureok
```

* */usr/share/courier/webadmin/webadmin.pl*

33.15. Komplexní instalace Postfix+Dovecot

http://www.debianadmin.com/debian-mail-server-setup-with-postfix-dovecot-sasl-squirrel-mail.html

```
# aptitude install postfix postfix-tls sasl2-bin libsasl2-modules popa3d
```

```
# aptitude install dovecot-imapd dovecot-pop3d dovecot-common
```

33.16. Domain Technologie Control (DTC)

```
$ aptitude search dtc
```

Kapitola 34. NFS (Network File System)

34.1. Sprovoznění jaderného NFS serveru

Nainstalujeme

```
# apt-get install nfs-kernel-server
```

A nakonfigurujeme, což znamená pouze vyexportovat adresáře které chceme mít přístupny po síti. Tento export je popsán v asresáři `/etc/exports`. Příklad exportu adresáře `/home/iowa` pro počítače `kvark` (čtení i zápis) a počítač `joshua` (pouze čtení):

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
/home/iowa      10.16.68.3(rw) 10.16.68.2()
```

34.2. Sprovoznění NFS serveru v uživatelském prostoru

FIXME: dopsat

34.3. Sprovoznění NFS klienta

Nainstlaujeme

```
# apt-get install nfs-common
```

Nyní se můžeme podívat jaké zdroje nám nabízejí jednotlivé nfs servery

```
kvark:~# showmount -e moon
Export list for moon:
/home/iowa joshua.firma.cz,kvark.firma.cz
kvark:~# showmount -e sunrise
Export list for sunrise:
/var/www 10.16.68.2
kvark:~#
```

Nyní obeznámeni s možnostmi si můžeme nějaké zdroje připojit. Například:

```
kvark:~# mount moon:/home/iowa /home/radek/iowa
```

Chceme-li připojovat tento síťový zdroj jednodušeji, pořídíme si pro něj záznam v tabulce `/etc/fstab`

```
# NFS svazky
moon:/home/iowa /home/radek/iowa nfs    noauto
```

34.4. Postupy

34.4.1. Jak

FIXME:

Kapitola 35. Samba

Instalace, konfigurace, poznámky, tipy

35.1. Tisk do „PDF tiskárny“

PDF tiskárna je virtuální zařízení jehož účelem je umožnit „tisk do PDF souboru“.

Nejdříve si vytvořím sdílení. Tedy v

1. .
2. Vytvořím si sdílení. Tedy vytvořím adresář

```
# mkdir /home/samba/pdffile
```

35.2. Nezpracované texty

35.2.1. Přidání stanice W2k do domény s PDC Samba

```
# adduser --gid 200 --uid 20004 --disabled-password --gecos "host dbxlyp0j" --home /dev/null -
Adding user dbxlyp0j...
Adding new user dbxlyp0j (20004) with group win2khost.
Home directory /dev/null already exists.  Not copying from /etc/skel
```

Kapitola 36. FTP

* *chapter id="ftp" xreflabel="FTP"*

V této kapitole se probírám protokolem FTP a několika servery pro něj.

Poznámky k chování ftp serveru.

FTP server by se měl řídit mimo jiné nastavením uživatelského shellu v souboru `/etc/passwd`. Jsou tři možnosti:

1. Uživatel má nastavený platný shell. V tom případě jsou mu služby ftp serveru dostupné.
2. Uživatel má nastavený platný shell `/bin/true`. V tom případš jsou mu služby ftp serveru dostupné.
3. Uživatel má nastavený neplatný shell `/bin/false`. Služby FTP serveru jsou mu odepřeny.

36.1. ProFTPd

* *section id="proftpd" xreflabel="ProFTPd"*

Odkazy a zdroje:

- ProFTPD (http://www.proftpd.org/)
- Odkaz ()

FTP server ProFTPd

36.1.1. Virtuální účty

* *Odzkoušeno pod Etch a Lenny*

Nejdříve si musíme připravit skupinu a uživatele (jednoho) pod jejichž identitou budou všichni virtuální uživatelé vystupovat. V případě jednoho systému již existovali (ftpgroup a ftpuser), jinak si potřebné vytvoříme.

```
# adduser --gid 999 --disabled-password --disabled-login customer
```

Pak si upravíme konfiguraci serveru v osuboru `/etc/proftpd/conf`. Najdeme a zaměníme, či přidáme následující direktivy:

```
DefaultRoot                     ~                           ❶
AuthUserFile                    /etc/proftpd/ftpd.passwd    ❷
DirFakeUser                     on      ~                   ❸
DirFakeGroup                    on      ~                   ❸
```

❶ Zajistí že každý uživatel uvidí svůj adresář jako kořenový. T.j. nemůže se podívat jinam.

❷ Umístění souboru se seznamem uživatelů ftp a jejich hesly.

❸ Tato dvě direktivy zajistí, že uživatel se vidí jako vlastník souborů. Je to aby nebyly nekteré programy zmateny.

Nyní máme vše připraveno pro vytváření uživatelů. Každému uživateli vytvoříme domovský adresář který bude vlastnit výše uvedený uživatel `ftpuser` ze skupiny `ftpgroup`. Například chceme založit virtuánlího uživatele `korek`, který si bude spravovat web v adresáři `/var/www/korkovo`. Vytvoříme si teda adresář.

```
# mkdir /var/www/korkovo
```

Nezapomeneme nastavit vlastníka tohoto adresáře.

```
# chown ftpuser:ftpgroup /var/www/korkovo
```

A na závěr vytvoříme našemu virtuálnímu uživateli účet s heslem

```
# ftpasswd --passwd --name=korek --uid=1004 --gid=108 \
 --home=/var/www/korkovo --shell=/bin/bash --file=/etc/proftpd/ftpd.passwd
```

Části `--uid=1004` a `--gid=108` identifikují našeho ftpuser a ftpgroup.

36.1.2. Konfigurační direktivy

V této části popisuji některé z konfiguračních direktiv. Jejich úplný seznam najdete v dokumentaci k programu.

Abecední seznam vybraných direktiv

- *CreateHome*
- .

36.1.2.1. mod_auth

* *section id="proftpd.mod_auth"*

FIXME:

CreateHome

Tato direktiva zajistí vytvoření a naplnění domovského adresáře uživatele v případě že tento neexistuje. Její použití je následující:

CreateHome [off|on [*`mode`* [skell *`path`*] [dirmode *`mode`*]]]

Poznámka: Direktiva je implementována až od verze 1.2.10.

36.1.2.2. mod_wrap -- Interface to libwrap

* *section id="proftpd.mod_wrap"*

TCPAccessFiles TCPAccessSyslogLevels TCPGroupAccessFiles TCPServiceName TCPUserAccessFiles

`TCPAccessFiles`

FIXME:

`TCPAccessSyslogLevels`

FIXME:

`TCPGroupAccessFiles`

FIXME:

`TCPServiceName`

FIXME:

`TCPUserAccessFiles`

FIXME:

server-wide access files
TCPAccessFiles /etc/ftpd.allow /etc/ftpd.deny
per-user access files, which are to be found in the user's home directory
TCPAccessFiles ~/my.allow ~/my.deny

36.1.3. Příklad instalace na serveru `wufi`

Nejdříve instalace balíčku

```
wufi:~# apt-get install proftpd
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  proftpd-common
The following NEW packages will be installed:
  proftpd proftpd-common
0 packages upgraded, 2 newly installed, 0 to remove and 0  not upgraded.
Need to get 250kB of archives. After unpacking 721kB will be used.
Do you want to continue? [Y/n] Y

ProFTPd configuration
 You did not install any debconf-enabled versions of the proftpd yet.
This means that I shall ask you some questions about the way to start
proftpd and anonymous access.

However, since you had proftpd installed before, I need to know if it
is ok for me to edit  to actually make those changes.  Only answer yes
if you did not change much.  You will have a chance to review the
changes later on.

If you answer 'no' here, I will not ask questions that may require me
to edit the file.

Edit configuration file ?  Yes

qqqqqqqqqqqqqqqqqqqqqqqqu ProFTPd configuration tqqqqqqqqqqqqqqqqqqqqqqqqqqk
  x ProFTPd can be run either as a service from inetd, or as a standalone    x
  x server.  Eeach choice has its own benefits.  If you have only a few ftp  x
  x connections per day, it may not be worth the resources to keep proftpd   x
  x running at all times.                                                    x
  x                                                                          x
  x On the other hand, if your ftp site is visited frequently, inetd is not  x
  x a good choice, because each time a connection is opened, a new process   x
  x is spawned.                                                              x
  x                                                                          x
  x Run proftpd from inetd or standalone?                                    x
  x                                                                          x
  x                              standalone                                  x
```

```
  x                              inetd
inetd

qqqqqqqqqqqqqqqqqqqqqqqu ProFTPd configuration tqqqqqqqqqqqqqqqqqqqqqqqqk
   x                                                                   x
   x Enabling anonymous access grants access to your ftp site by any user,  x
   x without being a system user.  This is probably not a good idea for     x
   x paranoid sysadmins.  Only enable if you know what it does.             x
   x                                                                   x
   x Enable anonymous access?                                          x
   x                                                                   x
   x                   <Yes>                      <No>                 x
   x                                                                   x
   mqqqqq
No

   lqqqqqqqqqqqqqqqqqqqqqqqu ProFTPd configuration tqqqqqqqqqqqqqqqqqqqqqqqqk
   x                                                                   x
   x I have made some changes to /etc/proftpd.conf; they are now in         x
   x /etc/proftpd.conf.proftpd-new. I saved the old version to              x
   x /etc/proftpd.conf.proftpd-old.  Do you want me to replace the current  x
   x configuration with my changes now, or do you want to review            x
   x /etc/proftpd.conf.proftpd-new and merge the changes yourself?          x
   x                                                                   x
   x Shall I modify the /etc/proftpd.conf automatically?                    x
   x                                                                   x
   x Do you want /etc/proftpd.conf to be updated?                           x
   x                                                                   x
   x                   <Yes>                      <No>                 x
   x                                                                   x
   mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
Yes
```

Konfigurační soubor pak upravím na:

```
ServerName                      "Wufi.Giga-Net.CZ
ServerType                      inetd
DeferWelcome                    off

ShowSymLinks                    on
MultilineRFC2228                on
DefaultServer                   on
ShowSymlinks                    on
AllowOverwrite                  on

TimeoutNoTransfer               600
TimeoutStalled                  600
TimeoutIdle                     1200

DisplayLogin                    welcome.msg
DisplayFirstChdir               .message
LsDefaultOptions                "-l"

DenyFilter                      \*.*/

Port                            21

MaxInstances                    5
```

```
IdentLookups                    on
SyslogLevel                     debug

AuthUserFile                    /etc/ftpd.passwd
```

Vytvořím účet pro užtivatele

```
wufi:~# ftpasswd --passwd --name=dusan --uid=33 --gid=33 \
                 --home=/var/www --shell=/bin/false --file=/etc/ftpd.passwd
```

36.1.4. Různé příklady a ukázky

36.1.4.1. Jemné řízení nastavení chrootu

U příkazu/direktivy DefaultRoot můžeme jako druhý parametr použít podmínku příslušnosti (`staff`) či nepříslušnosti (`!staff`) do skupiny. Tím můžeme řídit uplatnění DefaultRoot příkazu jak je vidět na následující ukázce.

```
# A more complex setup where all users are locked into
# their home except those in group 'staff' who are
# locked into /u2/allweb
#
<VirtualHost myhost.mynet.foo>
DefaultRoot ~ !staff
DefaultRoot /u2/allweb staff
</VirtualHost>
```

Vypsáno z: Finer grained control (http://proftpd.linux.co.uk/localsite/Userguide/linked/x751.html).

36.1.4.2. Automatické vytváření domovských adresářů

Pro automatické zakládání domovských adresářů na ftp serveru slouží direktiva *CreateHome*. Tuto direktivu ale zná ProFTPd až od verze 1.2.10. Protože jsem nutně potřeboval nějak vytvořit domovské účty k adresářům které byly omylem smazány, napsal jsem si jednoduchý skript. Uvádím jej bez úprav tak jak jsem jej použil.

```
#!/bin/sh
# Znovuvytvoření domovský adresářů ftp účtů
# Copyright (c) 2005 Radek Hnilica

# Tento skript je určen k opětovnému vytvoření domovských adresář ftp
# účtů které byly omylem smazány při špatně nastavené minimální hloubce
# mazání ve skriptu /root/bin/purge

# Soubor s účty
PASSWD=/etc/ftpd.passwd

cat $PASSWD| while IFS=':' read name pass uid gif gecos home shelll; do
        if [ ! -d $home ]; then
                echo "Creating home for $name. $home"
                mkdir -p $home
                chown customer:customer $home
                chmod 700 $home
        fi
done
```

FIXME: Popsat užití direktivy *CreateHome* až ji někde nasadím.

36.2. WU-FTP

FIXME:

/etc/ftpaccess
/etc/ftphosts
/etc/ftpusers -- !! uživatelé kteří nemají právo používat FTP

Kapitola 37. Internet Relay Chat

Šablona pro nové kapitoly

* *chapter id="fragment.irc" condition="author"*

37.1. IRC klienti

Já osobně používám už nějakou dobu XChat.

37.1.1. IRSSI

Odkazy:

- How to use screen and irssi (http://lizzie.spod.cx/screenirssi.shtml)
-

IRSSI je terminálový IRC klient. Tedy klient který běží v alfanumerickém terminálu a nepotřebuje k provoz X-Server.

37.2. Boti

Odkazy a zdroje:

- bobot++ (http://unknownlamer.org/code/bobot.html)
- Bobot++ (http://savannah.nongnu.org/projects/bobotpp/)

37.2.1. plum

IRC bot psaný v Javě

37.2.2. madoka

Odkazy a zdroje:

- The madoka project (http://www.madoka.org)

IRC personal proxy server, stationing on the IRC net with logging. and some bot plugins included

K 2003-09-04 jsou v distribuci debianu dostupné balíčky `madoka 4.2.3-1`(stable), `madoka 4.2.7-1`(testing, unstable).

Dokumentace k programu madoka je téměř výhradně v japonštině.

Po spuštění čte madoka obsah proměnné prostředí `$MADOKADIR` a v zde uvedeném adresáři hledá konfigurační soubor `madoka.rc`. Pokud není tato proměnná definována prohledává se adresář `$HOME/.madoka`.

Přepínače na příkazové řádce

`-rc` *`rcfile`*

Konfigurační soubor `madoka.rc`.

`-modes` *`modesfile`*

Konfigurační soubor `madoka.modes`.

`-nofork`

Nespouští se na pozadí.

37.2.3. infobot

Odkazy a zdroje:

- infobot — The bot with brains and good looks (http://www.infobot.org)

Zpravuje databázi faktů. K dispozici jsou již připravené a celkem veliké databáze faktů.

37.2.4. blbootbot

Odkazy a zdroje:

- Bloot Bot — The slowest and most bloated bot in the world (http://blootbot.sourceforge.net)

K 2003-09-04 jsou v distribuci debianu dostupné balíčky `blootbot 1.1.0-2`(*stable*), `blootbot 1.1.0-5`(*testing, unstable*).

37.2.5. Nezpracované poznámky

```
plum    IRC proxy, stationing, logging, bot, pirc
madoka  IRC personal proxy, stationing, logger and bot program (pirc)
muh     Full-featured IRC bouncing tool.
blootbot       a severely modified infobot for IRC
bnc     IRC Session Bouncing Proxy
bobot++ An IRC bot with scripting features
dircproxy       IRC proxy for people who use IRC from different workstation
ezbounce        A highly configurable IRC proxy


= dircproxy
Při instalaci se nepřiinstaluje žádný další balíček.
Nenašel jsem v standardní instalaci nic co by mohlo fungovat jako bot.  Asi jsou
 to skripty do dircproxy.

=
bobot++
Při instalaci si doinstaluje
         bobot++ libguile9 libltdl3
```

Kapitola 38. rsync server a klient

Efektivní zrcadlení souborů z jednoho stroje na jiný

Zrcadlo, zrcadlo, kdo je na světě nejrychlejší.

38.1. Sprovoznění rsync serveru

Pro uvedení rsync serveru do provozu a publikování části adresářové struktury musíme učinit následující:

1. Samozřejmě nainstalovat program (balíček) rsync
2. Do souboru `/etc/services` přidáme řádek

   ```
   rsync   873/tcp
   ```

   abychom se na port 873 mohli odkazovat symbolickým jménem rsync.
3. Budeme-li spouštět rsync pomocí inetd, vložíme do souboru `/etc/inetd.conf` řádku

   ```
   rsync      stream      tcp         nowait      root     /usr/bin/rsync rsyncd --daemon
   ```

   Jestli jej budeme spouštět jinak, tento krok vypustíme.
4. Poté si vytvoříme soubor, například `/root/rsyncd.conf` popisující adresáře jenž zpřístupníme a podmínky za jakých je tak učiněno.

   ```
   [mirror]
       path = /mobile/mirror
       comment = Původní mirror řady různých věcí
       read only = true
       host allow = ferit.moraviapress.cz
       max connections = 1
       uid = 0
       use chroot = true
   ```
5. A když už máme server nakonfiurovaný, můžeme ho rovnou spustit:

   ```
   joshua:~# rsync --daemon --config=/root/rsyncd.conf
   ```

38.2. Kopie disku

```
# rsync -avxrP --delete $FILESYSTEMS backup-server:backups/$HOSTNAME

Some caveats if you want to fully automate this...
  - remove -vP (verbose w/ progress)
  - --delete is NECESSARY to make sure deleted files get deleted from the
    backup
  - FILESYSTEMS should be any local filesystems you want backed up (-x
    won't cross filesystems, makes backing up in NFS environment easier)
  - obviously this doesn't preclude a bad guy checking out
    backup-server:backups/otherhostname (use ssh keys, and invoke cmd="cd
    backups/hostname; rsync with whatever daemon options" will limit that)
  - on backup-server, rotate the backup every 12 hours or whatever.
       - rsync -ar --delete store/hostname.2 store/hostname.3
```

```
        - rsync -ar --delete store/hostname.1 store/hostname.2
        - rsync -ar --delete backups/hostname store/hostname.1
        # that could be better optimized, but you get the idea
```

I’ve used this rsync system to successfully maintain up to date backups w/ great ease, AND restore very quickly... use a LinuxCare Bootable Business Card to get the target fdisked and ready, then mount the filesystems as you desire, and rsync -avrP backup-server:backups/hostname /target. I got a 700mb server back online in under 20 minutes from powerup to server serving requests (the rsync itself is 3 to 5 minutes). Making sure you do (cd /target; lilo -r . -C etc/lilo.conf)) is the only tricky part.

Kapitola 39. LDAP

Odlehčené adresářové služby, instalace, správa, použití

* *chapter id="chapter.ldap"*

Popis kapitoly.

```
LDAP protokol
Sprovoznění serveru
Sprovoznění klienta
sprovoznění replikace
```

Zdroje a odkazy.

- A System Administrator's View of LDAP (http://people.netscape.com/bjm/whyLDAP.html) by Bruce Markey
- LDAP guru <DOT> com (http://www.ldapguru.com/)

Seznam programů. pro práci s LDAP databází

- vlad - LDAP visualisation tool

39.1. Co je to LDAP

LDAP neboli *Lightweight Directory Access Protocol* je standardní protokol pro přístup ka adresářovým službám X.500. Protokol běží nad internetovými transportními protokoly jako je TCP.

LDAP je „odlehčená" alternativa X.500 *Directory Access Protocol* (DAP) určená pro použití na internetu. Doporučení X.500 jsou přístupná u ITU (http://www.itu.int).

Tabulka 39-1. Seznam RFC dokumentujících LDAPv2

RFC	název
RFC1777 (http://www.rfc-editor.org/rfc/rfc1777.txt)	Lightweight Directory Access Protocol
RFC1778 (http://www.rfc-editor.org/rfc/rfc1778.txt)	The String Representation of Standard Attribute Syntaxes
RFC1779 (http://www.rfc-editor.org/rfc/rfc1779.txt)	A String Representation of Distinguished Names
RFC1960 (http://www.rfc-editor.org/rfc/rfc1960.txt)	A String Representation of LDAP Search Filters

Tabulka 39-2. Seznam RFC dokumentujících LDAPv3

RFC	název
RFC2251 (http://www.rfc-editor.org/rfc/rfc2251.txt)	Lightweight Directory Access Protocol (v3)

RFC	název
RFC2252 (http://www.rfc-editor.org/rfc/rfc2252.txt)	LDAPv3: Attribute Syntax Definitions
RFC2253 (http://www.rfc-editor.org/rfc/rfc2253.txt)	LDAPv3: UTF-8 String Representation of Distinguished Names
RFC2254 (http://www.rfc-editor.org/rfc/rfc2254.txt)	The String Representation of LDAP Search Filters
RFC2255 (http://www.rfc-editor.org/rfc/rfc2255.txt)	The LDAP URL Format
RFC2256 (http://www.rfc-editor.org/rfc/rfc2256.txt)	A Summary of the X.500(96) User Schema for use with LDAPv3
RFC2829 (http://www.rfc-editor.org/rfc/rfc2829.txt)	Authentication Methods for LDAP
RFC2830 (http://www.rfc-editor.org/rfc/rfc2830.txt)	LDAPv3: Extension for Transport Layer Security
RFC3377 (http://www.rfc-editor.org/rfc/rfc3377.txt)	Lightweight Directory Access Protocol (v3): Technical Specification

39.2. Instalace

Pro práci s LDAP serverem je treba si nainstalovat nástroje. Na Debian Potato je to balíček umich-ldap-utils. Při instalaci přímo na serveru pak v konfiguraci ponecháme `localhost` jako defaultní LDAP server.

```
moon:~# wajig install slapd
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  slapd
0 packages upgraded, 1 newly installed, 0 to remove and 0  not upgraded.
Need to get 0B/607kB of archives. After unpacking 1810kB will be used.
Preconfiguring packages ...
Selecting previously deselected package slapd.
(Reading database ... 16989 files and directories currently installed.)
Unpacking slapd (from .../slapd_2.0.23-6_i386.deb) ...
Setting up slapd (2.0.23-6) ...
start-stop-daemon: warning: failed to kill 3797: No such process
Starting OpenLDAP: slapd slurpd.

moon:~#
```

Přinstalujeme nástroje a další programy

```
# wajig install libpam-ldap postfix-ldap
```

Seznam balíčků

- `slapd` — OpenLDAP server
- `scalemail` — Scalable virtual mail domain system built on Postfix and LDAP
- `ldapexplorer`
- `libpam-ldap`
- `ldap2dns`

- `ldap-utils`
- `ldaptor-utils`
- `ldaptor-webui`
- `libldap-ruby`

Vybrané soubory

- `/etc/ldap/scheme/scalemail.schema` — LDAP schéma z balíčku `scalemail`

39.3. Replikace LDAP databáze

Master a slave LDAP

39.3.1. Nastavení „master“ serveru

Do souboru `/etc/openldap/slapd.conf` přidáme:

```
# Konfigurace replikování na ldap-slave
replica host=ldap-slave.domain.cz \
        binddn="uid=ldap-replicator,ou=People,dc=domain,dc=cz" \
        bindmethod=simple credentials=secret
```

39.3.2. Nastavení „slave“ serveru

empty

39.3.3. Obnovení repliky.

Obnovení stavu *slave* z *master* serveru

Postup

- Zastavíme všechny LDAP servery *master* i *slave*

```
ldap-master:~# /etc/init.d/openldapd stop
ldap-slave:~# /etc/init.d/openldapd stop
```

- Zkopírujeme obsah adresáře `/var/lib/openldap` z *master* serveru do stejného adresáře na *slave* serveru

```
ldap-master # tar cf - /var/lib/openldap|gzip -9|\
        ssh ldap-slave "cd /; tar xzf -"
```

- Opětovně uvedem LDAP servery do provozu

```
ldap-slave:~# /etc/init.d/openldapd start
ldap-master:~# /etc/init.d/openldapd start
```

Vytvořil jsem si malý skript, který toto automatizuje

Příklad 39-1. Skript pro obnovení slave LDAP serveru

```
#!/bin/sh
# $Header: /home/radek/cvs/unix-book/input/unix/ch-ldap.xml,v 1.1.1.1 2009-01-24 15:42:51 rade
# Obnoveni stavu ldap repliky.  Nasilna cesta.
# Copyright (C) 2001 Radek Hnilica
# All rights reserved.

SLAVE=ldap-slave-pha

/etc/init.d/openldapd stop
tar cf - /var/lib/openldap|gzip -9| ssh $SLAVE "
    /etc/init.d/openldapd stop
    cd /
    tar xzf -
    /etc/init.d/openldapd start
    "
/etc/init.d/openldapd start
```

39.4. Testování

Základní test funkčnosti je prohledávání

```
$ ldapsearch -h ldap.domain.cz -b "dc=domain,dc=cz" \
> 'objectClass=*'


$ ldapsearch -h ldap.domain.cz -b "dc=domain,dc=cz" \
> 'uid=userid_uživatele'
```

39.5. Správa LDAP databáze

Jak spravovat LDAP databázi. Jak založit uživatele, změnit, ...

39.5.1. Založení uživatele

```
   $ cat ->file
   dn: uid=jiriv,ou=People,dc=firma,dc=cz
   objectClass: top
 5 objectClass: person
   objectClass: account
   cn: Jiri Vomacka
   givenName: Jiri
   sn: Vomacka
10 uid: jiriv
```

```
    mailDrop: jiriv@moraviapress.cz
    mailAcceptingGeneralId: Jiri.Vomacka
    userPassword: tajne-heslo

15
```

nyní připravené informace nahrajeme

```
#!/bin/sh
LDAPBASE="dc=firma,dc=cz"
PEOPLE="ou=People,$LDAPBASE"
ADMIN="uid=admin,ou=People,$LDAPBASE"

ldapdelete -h ldap -D $ADMIN -w 'tajne heslo' "uid=jiriv,$PEOPLE"
ldapadd -h ldap -D $ADMIN -w 'tajne heslo' -f file
```

39.5.2. Změna hesla

```
$ ldappasswd -h ldap -D "uid=uživatel,ou=People,dc=firma,dc=cz" -W
```

39.6. Udržování záznamů DNS v LDAP databázi

- LDAP Implementation HOWTO - DNS (http://www.linuxdoc.org/HOWTO/LDAP-Implementation-HOWTO/dns.html)

LDAP to DNS gateway (http://ldap2dns.tiscover.com)From the site:

ldap2dns is a program to create DNS (Domain Name Service) records directly from a LDAP directory. It can and should be be used to replace the secondary name-server by a second primary one.

ldap2dns reduces all kind of administration overhead: No more flat file editing, no more zone file editing. After having installed ldap2dns, the administrator only has to access the LDAP directory.

Optionally she can add access control for each zone, create a GUI and add all other kind of zone and resource record information without interfering with the DNS server.

ldap2dns is designed to write ASCII data files used by tinydns from the djbdns package, but also may be used to write .db-files used by named as found in the BIND package.

39.7. Schema

Odkazy / Zdroje:

- RFC2307: An Approach for Using LDAP as a Network Information Service (http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2307.html)

- OpenLDAP OID Registry (http://www.openldap.org/faq/index.cgi?file=197)
- Object Identifiers (http://www.alvestrand.no/harald/objectid/)

> **Varování**
>
> Under no circumstances should you use a fictious OID!

.

```
nisNetgroupTripleSyntax ::= SEQUENCE {
    hostname    [0] IA5String OPTIONAL,
    username    [1] IA5String OPTIONAL,
    domainname  [2] IA5String OPTIONAL
}
```

39.7.1. Rozšíření schemat

Odkazy / Zdroje:

- Schema Specification (http://www.openldap.org/doc/admin20guide.html#Schema%20Specification)

Dodaná schémata si můžeme rozšířit o nové pravidla, atributy a třídy.

Pět kroků k vytvoření nového schematu:

1. získat identifikátor objektu *Object Identifier*
2. vybrat si prefix pro jména *name prefix*
3. vytvořit lokální soubor se schématem
4. definovat vlastní atributy typů (je-li to potřeba)
5. definovat vlastní třídy objektů *Object Classess*

39.7.2. Atributy

Direktiva `attributeType` definuje nové typy atributů. Její syntaxe je posána v RFC2252 (http://www.rfc-editor.org/rfc/rfc2252.txt)

```
attributeType <RFC2252 (http://www.rfc-editor.org/rfc/rfc2252.txt) Attribute Type Description>
```

kde Attribute Type Description je definován následujícím BNF:

```
AttributeTypeDescription = "(" whsp
        numericoid whsp                 ; AttributeType identifier
      [ "NAME" qdescrs ]                ; name used in AttributeType
      [ "DESC" qdstring ]               ; popis -- description
      [ "OBSOLETE" whsp ]
      [ "SUP" woid ]                    ; odvozeno z atributu woid
      [ "EQUALITY" woid ]               ; Matching Rule Name
      [ "ORDERING" woid ]               ; Matching Rule Name
      [ "SUBSTR" woid ]                 ; Matching Rule Name
      [ "SYNTAX" whsp noidlen whsp ]    ; viz Podporované syntaxe
      [ "SINGLE-VALUE" whsp ]           ; default multi-valued
      [ "COLLECTIVE" whsp ]             ; default not collective
      [ "NO-USER-MODIFICATION" whsp ]   ; default user modifiable
      [ "USAGE" whsp AttributeUsage ]   ; default userApplications
      whsp ")"
```

```
AttributeUsage =
        "userApplications"      /
        "directoryOperation"    /
        "distributedOperation"  / ; DSA-shared
        "dSAOperation"            ; DSA-specific, value depends on server
```

39.7.3. Identifikátory objektů (OID)

Každý prvek schématu je jednoznačně určen globálním OID (*Object IDentifier*). OID jsou také použity pro identifikaci jiných objektů. OID mají hierarchickou strukturu a jsou unikátní na celém světě. Tato unikátnost je zaručena centrální autoritou, která přiděluje prefixy OID.

Tabulka 39-3. Příklad hierarchie OID

1.1	OID organizace
1.1.1	SNMP Elements
1.1.2	LDAP Elements
1.1.2.1	AttributeTypes
1.1.2.1.1	myAttribute
1.1.2.2	ObjectClasses
1.1.2.2.1	myObjectClass

OID si můžete nechat přidělit/zaregistrovat zdarma u Internet Assigned Numbers Authority (http://www.iana.org/). Každá soukromá organizace může podat žádost o OID. Učiná tak vyplněním formuláře http://www.iana.org/cgi-bin/enterprise.pl. Přidělená čísla/prefixy mají tvar `1.3.6.1.4.1.x` kde `x` je celé číslo.

39.7.4. Specifikace Typu Atributů

Například typy atributů `name` a `cn` jsou v souboru `core.schema` definovány takto:

```
attributeType (2.5.4.41 NAME 'name'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{332768} )
attributeType ( 2.5.4.3 NAME
        ( 'cn' $ 'commonName' ) SUP name )
```

Tabulka 39-4. Podporované syntaxe

název	**OID**	**popis**
binary	1.3.6.1.4.1.1466.115.121.1.5	BER/DER data
boolean	1.3.6.1.4.1.1466.115.121.1.7	boolean value
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	DN
directoryString	1.3.6.1.4.1.1466.115.121.1.15	UTF-8 string

název	OID	popis
IA5String	1.3.6.1.4.1.1466.115.121.1.26	ASCII string
Integer	1.3.6.1.4.1.1466.115.121.1.27	integer
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	DN plus UID
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	numeric string
OID	1.3.6.1.4.1.1466.115.121.1.38	object identifier
Octet String	1.3.6.1.4.1.1466.115.121.1.40	arbitrary octets
Printable String	1.3.6.1.4.1.1466.115.121.1.44	printable string

Tabulka 39-5. Podporovaná srovnávací pravidla

název	typ	popis
booleanMatch	equality	boolean
objectIdentifierMatch	equality	OID
distinguishedNameMatch	equality	DN
uniqueMemberMatch	equality	DN with optional UID
numericStringMatch	equality	numerical
numericStringOrderingMatch	ordering	numerical
numericStringSubstringsMatch	substrings	numerical
caseIgnoreMatch	equality	case insensitive, space insensitive
caseIgnoreOrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreSubstringsMatch	substrings	case insensitive, space insensitive
caseExactMatch	equality	case sensitive, space insensitive
caseExactOrderingMatch	ordering	case sensitive, space insensitive
caseExactSubstringsMatch	substrings	case sensitive, space insensitive
caseIgnoreIA5Match	equality	case insensitive, space insensitive
caseIgnoreIA5OrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreIA5SubstringsMatch	substrings	case insensitive, space insensitive
caseExactIA5Match	equality	case sensitive, space insensitive
caseExactIA5OrderingMatch	ordering	case sensitive, space insensitive
caseExactIA5SubstringsMatch	substrings	case sensitive, space insensitive

Řada příkladů

```
attributeType ( 1.3.6.1.4.1.4203.666.1.5
        NAME 'OpenLDAPaci'
        EQUALITY OpenLDAPaciMatch
        SYNTAX 1.3.6.1.4.1.4203.666.2.1
        USAGE directoryOperation
```

```
# Pseudo-attribute type used internally by OpenLDAP 2.0
attributeType ( 1.3.6.1.4.1.4203.1.3.1
        NAME 'entry'
        DESC 'OpenLDAP ACL entry pseudo-attribute'
        SYNTAX 1.3.6.1.4.1.4203.1.1.1
        SINGLE-VALUE NO-USER-MODIFICATION USAGE dSAOperation )


# Pseudo-attribute type used internally by OpenLDAP 2.0
attributeType ( 1.3.6.1.4.1.4203.1.3.2
        NAME 'children'
        DESC 'OpenLDAP ACL children pseudo-attribute'
        SYNTAX 1.3.6.1.4.1.4203.1.1.1
        SINGLE-VALUE NO-USER-MODIFICATION USAGE dSAOperation )


# Delegated for use in RFC3112 (http://www.rfc-editor.org/rfc/rfc3112.txt)
attributeType ( 1.3.6.1.4.1.4203.1.3.3
        NAME 'supportedAuthPasswordSchemes'
        DESC 'supported password storage schemes'
        EQUALITY caseExactIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{32}
        USAGE dSAOperation )


# Delegated for use in RFC3112 (http://www.rfc-editor.org/rfc/rfc3112.txt)
attributeType ( 1.3.6.1.4.1.4203.1.3.4
        NAME 'authPassword'
        DESC 'password authentication iformation'
        EQUALITY 1.3.6.1.4.1.4203.1.2.2
        SYNTAX 1.3.6.1.4.1.4203.1.1.2 )


attributeType ( 1.1.2.1.1
        NAME 'myUniqueName'
        DESC 'unique name with my organization'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )


attributeType ( 1.1.2.1.1
        NAME 'myUniqueName'
        DESC 'unique name with my organization'
        SUP name )
```

39.8. Třídy objektů

Direktiva `objectClass` se používá pro definování nových tříd objektů.

```
objectClass <RFC2252 (http://www.rfc-editor.org/rfc/rfc2252.txt) Object Class Description>
```

kde Object Class Description je definována následujícím BNF:

```
ObjectClassDescription = "(" whsp
```

```
        numericoid whsp         ; ObjectClass identifier
        [ "NAME" qdescrs ]
        [ "DESC" qdstring ]
        [ "OBSOLETE" whsp ]
        [ "SUP" oids ]          ; Superior ObjectClasses
        [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
                                ; default structural
        [ "MUST" oids ]         ; AttributeTypes
        [ "MAY" oids ]          ; AttributeTypes
        whsp ")"
```

39.8.1. Příklady

```
objectClass ( 1.1.2.2.1
        NAME 'myPhotoObject'
        DESC 'mixin myPhoto'
        AUXILIARY
        MAY myPhoto )


objectClass ( 1.1.2.2.2
        NAME 'myPerson'
        DESC 'my person'
        MUST ( 'myUniqueName' $ 'givenName' )
        MAY 'myPhoto' )
```

39.9. OID Atributy

Tabulka 39-6. OID

1.3.6.1.4.1.1466.115.121.1.15{40}	FIXME:
1.3.6.1.4.1.8833	ISPMan's Assigned Base OID
1.3.6.1.4.1.8833.1	SNMP Elemets
1.3.6.1.4.1.8833.2	LDAP Elemets
1.3.6.1.4.1.8833.2.1	AttributeTypes
1.3.6.1.4.1.8833.2.1.1299	ObjectClass ispmanMailGroup
1.3.6.1.4.1.8833.2.2	ObjectClasses

39.10. Některé třídy a jejich použití

39.10.1. Třída person

Záznam o osobě.

```
objectClass ( 2.5.6.6
        NAME 'person'
        SUP top STRUCTURAL
        MUST ( sn $ cn )
        MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

39.10.2. Třída `organizationalPerson`

```
objectClass ( 2.5.6.7
        NAME 'organizationalPerson'
        SUP person STRUCTURAL
        MAY ( title $ x121Address $ registeresAddress $ destinationIndicator $
              preferredDeliveryMethod $ telexNumber $ telexTerminalIdentifier $
              telephoneNumber $ internationaliSDNNumber $
              facsimileTelephoneNumber $ street $ postofficeBox $ postalCode $
              postalAddress $ physicalDeliveryOfficeName $ ou $ st $ l ) )
```

39.10.3. OpenLdap Person

```
objectClass ( 1.3.6.1.4.1.4203.1.4.5
        NAME 'OpenLDAPperson'
        DESC 'OpenLDAP Person'
        SUP ( pilotPerson $ inetOrgPerson )
        MUST ( uid $ cn )
        MAY ( givenName $ labeledURI $ o ) )
```

39.11. Nezpracované texty

39.11.1. LDAP záznamy pro ISPMan

Tabulka 39-7. Komentovaná ukázka

user	agh	uživatelské jméno, účet
mailLocalAddress	agh@firma.org	oficiální e-mail adresa
mailRoutingAddress	agh@server10.firma.org	server s poštovní přihrádkou uživatele

mailForwardingAddress	agh@developer.cz	přesměrování pošty
mailForwardingAddress	atif@jinafirma.com	další přesměrování

V ukázce je dopis doručen na `agh@firma.org` (interní poštovní úřad/sklad) a kopie jsou poslány na `agh@developer.cz` a `atif@jinafirma.com`.

Definované atributy a jejich význam

`mailLocalAddress`

Poštovní (email) adresa

`mailAlias`

přezdívky, další jména uživatele adresy

`mailRoutingAddress`

schránka uživatele

`mailForwardingAddress`

kam jinam se ještě posílá kopie příchozí pošty

39.12. Použití LDAPu ke konkrétním účelům

Zde popisuji navrhovaná či nasazená řešení jenž používají LDAP jako datový zdroj pro řadu různých účelů

39.12.1. Propojení pošty s LDAPem

Jak ukládat informace o poštovních schránkách do LDAPu.

Toto je nikoliv první řešení, ale přesto řešení minimální. Ukládám do LDAPu opravdu jen to nejnutnější co je třeba k funkčnosti propojení. Řadu problémů neřeším.

39.12.1.1. Návrh schématu

Prvím krokem k propojení je navrhnout schéma. Tedy jaké informace vlastně budeme do LDAPu ukládat. Použil jsem tyto atributy:

`mailLocalAddress` — elektronická adresa uživatele, hlavní adresa

(Odchozí adresa uživatele poštovní přihrádky) a současně jeho poštovní adresa (adresa přihrádky). Například: <`Kralik.Pokusny@Firma.CZ`> nebo <`Obchod@Firma.CZ`>. Atribut smí být pro každý LDAP záznam/uživatele použit jen jednou. To znamená že jeden uživatel má právě jednu poštovní adresu. Atribut je povinný.

`mailAlias` — přezdívka, aliasy poštovní adresy

Přezdívka (*alias*) je poštovní adresa přiřazená k záznamu/uživateli. Je to způsob jak přiřadit uživateli více adres. Použijeme jednu hlavní, a libovolný počt přezdívek. Přezdívky jsou vlastně jiné názvy pro tutéž poštovní přihrádku. Mají stejný tvar (strukturu) jako atribut `mailLocalAddress`. Atributu může být v záznamu libovolné množství a je nepovinný.

`mailRoutingAddress` — směrovací adresa

Tato velmi důležitá adresa je pro uživatele skryta. Poštovní systém ji však potřebuje, neb určuje skutečnou poštovní přihrádku do které mají být dopisy směrovány. V původním systému jsem ji nazýval `mailDelivery`. Sem je všemi MTA směrována pošta pro daného uživatele. Tento atribut umožňuje mít poštovní přihrádky části uživatelů na různých serverech. Tento atribut musí být v záznamu právě jeden. Atribut je povinný.

`mailForwardingAddress` — přeposílací adresy

V těchto atributech jsou zaznamenány adresy kam se zasílají kopie příchozí pošty. Záznamů může být libovolné množství nebo žádný. Uživatelům je umožněno je měnit a tím si nastavit například komu se bude zasílat kopie jejich pošty k vyřízení když budou na dovolené.

Varování

Je nutno vyřešit který poštovní počítač bude zodpovědný za tuto operaci, aby nedošlo k tomu, že kopie na stejnou adresu budou poslány vícekrát.

39.12.2. Ukládání informací o stanicích (DHCP,DNS)

Další informací kterou chci v LDAPu umístnit je základní popis technického vybavení pracovních stanic. Prvotním účelem je vést popis pracovních stanic a obzvláště informace potřebné pro sestavení konfiguračních souborů pro DHCP a DNS.

39.12.2.1. Návrh schématu

Rozpracovaný návrh

```
dn: host=kvark,ou=Hosts,dc=firma,dc=cz
objectClass: top
objectClass: host
host: kvark
macAddress: 00:11:22:33:44:55
ipv4Address: 10.22.33.44
description: obecný text
```

NIS ipHost Class

```
dn: cn=stroj.firma.cz,dc=firma,dc=cz
objectClass: top
objectClass: ipHost
```

`host` nebo `hostName`— jméno počítače/pracovní stanice

Název pracovní stanice. DNS jméno.

`macAddress` nebo `mac`

MAC adresa síťové karty.

`ipAddress` nebo `ipv4Address` nebo `ip`

IPv4 adresa stanice.

Kapitola 40. Čas

Přesný čas

* *chapter id="time"*
* *Popřemýšlet o rozčlenění na části a přidání do již existujících kapitol. Například věci kolem kompilace jádra by mohly být v části instalace a konfigurace.*

Přesný čas je nutností pro řadu aplikací, jedním z nejdůležitějsších momentů je přesný čas u záznamů v deníku. Bez něj nedáme dohromady záznamy z různých strojů a přesně nezjistíme kdy dané události proběhly.

40.1. `ntp` *Network Time Protocol*

V Debianu máme k dispozici několik balíčků:

- `ntp` — celý NTP server
- `ntp-simple` — jednoduchý NTP klient
- `ntpdate` — program pro jednorázové srovnání hodin

Časové servery k použití:

Tabulka 40-1. Časové servery

stratum	server
	`ntp.cesnet.cz`
	`ntp0.fau.de`
	`ntp1.ien.it`
	`ntp0.nl.net`
	`chime1.surfnet.nl`

`ntp` server a klient nainstalujeme jednoduše

```
# apt-get install ntp ntpdate
```

```
# wajig install ntp
```

Jen potřebujeme znát nějaké časové servery. Pár serverů zde uvedu.

Několik `ntp` serverů stratum 1

- `ntp.cesnet.cz`
- `ntp0.fau.de`
- `ntp1.ien.it`

Další servery je možno nalézti na Public NTP Time Servers (http://www.eecis.udl.edu/~mills/ntp/servers.htm)

40.1.1. Příklad Instalace na Debian Woody ve firmě

Na jedné ze satelitních sítí instaluji na centrálním serveru lokace časový server.

```
# apt-get install ntp ntpdate
```

Jako vnitrofiremní časové servery používám `ntp, ntp2, ntp3`. Tyto se zapíší do souboru `/etc/ntp.conf` pro démona `ntpd`.

```
server ntp
server ntp2
server ntp3
```

a do souboru `/etc/default/ntp-servers` pro program `ntpdate`

```
NTPSERVERS="ntp ntp2 ntp3"
```

40.2. Synchronizace času v síti

Jednotlivé počítače/hosty zasynchronizujeme pomcí `ntp` a `ntpdate`. Ntp slouží k průběžné synchronizaci času a ntpdate pak k jednorázové při startu stroje.

```
# aptitude install ntp-simple ntpdate
```

Po instalaci upravíme konfiguraci. Ntpdate nastavím v `/etc/default/ntpdate`. Zde v proměnné `NTPSERVERS` nastavíme naše časové servery.

```
NTPSERVERS="ntp.firma.cz"
```

Konfigurace ntp je pak v souboru `/etc/ntp.conf`

```
server ntp.firma.cz
```

V. Používání unixu a jeho nástrojů

Popis použití programů které se na UNIXu/Linuxu vyskytují.

Kapitola 41. Editory WORKING

* *chapter id="editory" xreflabel="Editory"*
* *print="**psselect -p358-369 unix.ps|foldprn -s12**"*

Kapitola 42. Záznam sezení

* *Tato kapitola je věnována programům pro záznam sezení jako jsou **script** a **ttyrec**. Posléze až vznike potřeba bych mohl popsat programy pro záznam událostí v GUI.*

V Linuxu máme k dispozici dva nástroje, které slouží pro záznam událostí na terminálu/konzoli. Jsou to **script** z balíčku bsdutils a **ttyrec** z balíčku ttyrec. Starším a původním je program **script**.

Jak program **script** funguje? Spustíme ho a jako parametr zadáme jméno souboru do kterého chceme uložit záznam sezení. Po spuštění se program připojí k terminálu a spustí nám shell ve kterém normálně pracujeme. Shell se chová standardně, ostatně tak jak očekáváme. Po ukončení pomocí příkazu **exit** nebo **Ctrl**+**d** se program ukončí a nám zůstane záznam sezení. V tomto souboru je uloženo vše co se na obrazovce terminálu dělo, je to taková hardcopy. Vytvořený soubor můžeme používat jako protokol o tom co jsme dělali.

```
$ script zaznam
Script started, file is zaznam
...
$ exit
Script done, file is zaznam
```

Pokud chceme více, musíme se záznamem sezení uložit i časové informace. Použijeme-li přepínač `-t`, posílá **script** tyto informace do standardího chybového výstupu. Program tedy vytvoří dva soubory. Soubor se záznamem toho co se dělo, a druhý soubor se záznamem jak dlouho to trvalo. Vzniklé soubory pak můžeme přehrát programem **scriptreplay**. Poslední parametr zadávaný **scriptreplay** znamená jakou rychlostí se má přehrávat. Jednička je stejná rychlost jakou byl záznam pořízen, dvojka dvojnásobná, trojka trojnásobná, atd.

```
$ script -t 2> sezeni2 sezeni2.timing
...
Script started, file is sezeni2
$ exit
Script done, file is sezeni2
$ scriptreplay sezeni2.timing sezeni2 2
```

Poznámka: Program zaznamenává dění na obrazovce v tom smyslu, že zaznamenává všechny znaky a řídící sekvence které jdou na terminál. Pokud tedy sezení používáme běžné programy, není problém. U celoobrazovkových programů, jako je například editor **vi** může nastat problém. Platí pravidlo, že sezení se musí přehrávat na stejném terminálu, myšleno co do vlastností a schopností, jako ten na kterém bylo zaznamenáno.

Alternativou k programům **script** a **scriptreplay** jsou programy **ttyrec** a **ttyplay** z balíčku ttyrec. Jejich chování a použití je obdobné. Hlavním rozdílem je to, že časování a události nejou uložené ve dvou samostatných souborech, nýbrž jen v jednom.

```
$ ttyrec sezeni3
...
$ ttyplay sezeni3
```

Kapitola 43. WWW Servery

* *chapter id="http"*

Odkazy:

- DEAD: Csáček 2.1(http://www.csacek.cz/) od Jaromíra Dolečka

Text kapitoly

43.1. Apache a čeština

Appache česky umí. tedy přesněji řečeno, pokud v hlavičce dokumentu specifikujete kódování, a váš prohlížeč jej podporuje tak je všechno v pořádku a nemáte problém. Tedy v praxi je to trochu jiné. Někdy se stránka zobrazí česky až na *reload* a jindy musíte ručně změnit kódování.

Ale existuje řešní, tedy řešení problému, který by neměl existovat. Jedná s o modul CSáček.

Instalace. je jednoduchá, do `/etc/apt/sources.list` jsem přidal řádek

```
deb ftp://debian/debian-czsk potato czech
```

nebo

```
deb ftp://ftp.debian.cz/debian-czsk potato czech
```

a do konfiguračního souboru Apache `/etc/apache/httpd.conf` přídáme nahrání modulu mod_csacek.so:

```
LoadModule csacek_module /usr/lib/apache/1.3/mod_csacek.so
```

do konfigurace virtuálního webu www.recom-obal.cz dvě direktivy

```
<VirtualHost 193.165.212.138>
        ServerName www.recom-obal.cz
        DocumentRoot /var/www/www.recom-obal.cz
        csacekEngine On
        csacekDefaultCharset windows-1250
</VirtualHost>
```

Jedná se o dvě posledí direktivy začínající „csacek“.

Seznam direktiv.

csacekBarDef
csacekChangeURL
...

43.2. Apache a ruština

Jiná cesta k národní podpoře.

Vyskytla se potřeba vystavit na webu jak české stránky, tak stránky v ruštině. Po nějaké době hraní s programem CSacek jsem zjistil, že tudy cesta nevede. Tedy aspoň mně se ji najít nepodařilo. Po dotazu v konferenci mi byl doporučen ruský apache (http://apache.lexa.ru/english/intro.html). Po odinstalování apache a nainstalování ze souborů na ftp://ftp.3logic.net/local/dists/potato/apache-rus/binary-i386/ a znovuvytvoření konfiguračních souborů zčala ruština a čeština bez problémů *automagicky* fungovat. Zatím jsem neprováděl v konfiguraci žádné změny ani jsem v nich nic důležitého neměnil. Tak jak se vytvořili jsem v nich opravil jen kořen webu. Ani

virtuální weby jsm zatím nenastavoval.

43.3. Další pokus s ruským apache v distribuci woody

Po upgrade stroje jenž poskytoval český a ruský web tento přestal chodit. Zkusil jse m tedy opětovne sprovoznit ruského apache. Do `/etc/apt/sources.list` jsem přidal řádku

```
# Binaries for Russian Apache
deb ftp://ftp.3logic.net/local woody apache-rus
```

Tento jse poté nainstaloval

```
# wajig update
# wajig install appache appache-doc
```

Nicméně pořád to bylo špatně. Zahloubal jsem se tedy do studia dokumentace. Podotazu v konferenci mi bylo doporučeno zkontrolovat volbu `AddDefaultCharset`. Tuto volbu jsem v souboru `httpd.conf` nastavil na

```
AddDefaultCharset Off
```

Po restartu Apache

```
# /etc/init.d/apache restart
```

a vyprázdnení vyrovnávacích pamětí v klientovi vše funguje na první pohled dobře. A to jak pod Mozillou 0.9.9 tak pod IE 5.50.4522.1800SP1

43.4. Analýza deníků Apache

Odkazy a zdroje:

- mergelog (http://awstats.sourceforge.net/)

ToDo

1. První úkol.

FIXME:

43.4.1. AWStats

* *http://awstats.sourceforge.net/*

Odkazy a zdroje:

- AWStats (http://awstats.sourceforge.net/)
- AWStats (http://www.awstats.org/)

ToDo

1. První úkol.

V Debian GNU/Linux jsou k dispozici tyto verze balíčku `awstats`

`4.0-0.woody.1` (woody)
`5.6-1` (sarge)
`5.6-1` (sid)

43.4.2. Analog

Odkazy a zdroje:

- Analog (http://www.analog.cx/)

V Debian GNU/Linux jsou k dispozici tyto verze balíčku `analog`

`2:5.23-0woody1` (woody)
`2:5.32-7` (sarge)
`2:5.32-8` (sid)

```
# apt-get install analog
```

43.4.3. Webalizer

V Debian GNU/Linux jsou k dispozici tyto verze balíčku `webalizer`

`2.01.10-2` (woody)
`2.01.10-19` (sarge)
`2.01.10-21` (sid)

43.5. Konfigurace Appache

FIXME:

43.6. Moduly k Apache

* *section id="apache-modules" xreflabel="Moduly k Apache" condition="author"*

43.6.1. `auth_ldap`

Seznam direktiv

- `AuthLDAPURL`
- `AuthLDAPBindDN`
- `AuthLDAPBindPassword`
- `AuthLDAPCompareDNOnServer`
- `AuthLDAPGroupAttribute`
- `AuthLDAPGroupAttributeIsDN`
- `AuthLDAPGroupAttributeIsDN`
- `require valid-user`
- `require user`
- `require dn`
- `require group`

43.6.1.1. Direktiva `AuthLDAPUrl`

```
protocol://server:port/base?attribute?scope?filter
```

```
ldaps://ldap.isrl.uiuc.edu/ou=People,dc=ISRL?uid?sub;ldaps://ldap2.isrl.uiuc.edu/
```

43.6.2. `auth-pam`

FIXME:

43.6.3. `auth-ldap`

Seznam direktiv

- `LDAPCacheEntries`
- `LDAPCacheTTL`
- `LDAPOpCacheEntries`
- `LDAPOpCacheTTL`
- `LDAPSharedCacheSize`
- `LDAPTrustedCA`
- `LDAPTrustedCAType`

Ukázková konfigurace

```
# Enable the LDAP connection pool and shared
# memory cache.  Enable the LDAP cache status
# handler.  Requires that mod_ldap and mod_auth_ldap
# be loaded.

LDAPSharedCacheSize 200000
LDAPCacheEntries 1024
LDAPCacheTTL 600
LDAPOpCacheEntries 1024
LDAPOpCacheTTL 600

<Location /ldap-status>
    SetHandler ldap-status
    Order deny,allow
    Allow from yourdomain.example.com
    AuthLDAPEnabled on
    AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one
    AuthLDAPAuthoritative on
    require valid-user
</Location>
```

43.6.3.1. Omezení přístupu ke stránkám

Omezení přístupu ke stránkám v nějakém podstromu hierarchie stránek zajistíme například takto:

```
### Omezení přístupu k dokumentu popisujícím strukturu sítě
<Location /document/sit-mpress>
```

```
        AuthLDAPEnabled on
        AuthLDAPAuthoritative   on
        AuthName        "LDAP autentikace"
        AuthType        Basic
        AuthLDAPURL     ldap://ldap/ou=people,dc=moraviapress,dc=cz?uid
        require         user radek honzat twada carlito pavelb jirig kamil
</Location>
```

Ve výše uvedeném příkladu použijeme jako zdroj ověření (autentikace) LDAP server `ldap`.

Varování

V modulu `auth_ldap` je zdá se chyba. Na řádku direktivy `require` nesmí být mezi klíčovým slovem `user` a uživatelem tabelátor. Mezera tam funguje.

43.7. Apache

* *section id="apache"*

43.7.1. Virtual web hosting

Virtual web hosting je technologie jak na jednom serveru hostovat více webů.

43.7.1.1. Name Based

.

Relevantni direktivy:

- DocumentRoot
- NameVirtualHost
- ServerAlias
- ServerName
- ServerPath
- VirtualHost

Příklad 43-1. Příklad konfigurace virtuálního web serveru

```
NameVirtualHost 213.29.4.4

<VirtualHost 213.29.4.4>
        ServerName www.giga-net.cz
        ServerPath /giga-net
        DocumentRoot /var/www/tcservis/giga-net
</VirtualHost>

<VirtualHost 213.29.4.4>
        ServerName www.mojenervy.cz
```

```
        ServerPath /mojenervy
        DocumentRoot /var/www/tcservis/mojenervy
</VirtualHost>
```

Poznámka: Místo ip adresy bachom mohli uvádět znak *, ale nejspíš protože provozuji celý www stroj jako virtuální počítač tak mi to nefunguje.

43.7.1.2. Úkoly / Tasky

FIXME:

43.7.1.2.1. Změna / nastavení hesla uživateli

Nastavení a změna hesla uživatele v soubor s hesly se provádí přikazem **htpasswd**. Nejdříve tedy vytvoření nového uživatele. To provádíme příkazem **htpasswd**. Zde je třeba dát pozor na drobnost. Přepínačem `-c` říkáme že se soubor s hesly má vytvořit. Tento přepínač použijeme tedy při zakládání prvního uživatele.

```
# htpasswd -c soubor-s-hesly uživatel
```

pokud soubor s hesly již existuje, nesmíme použít přepínač `-c` abychom si jej nesmazali. Změna hesla se provádí stejně jako při vytváření účtu, tedy

```
# htpasswd soubor-s-hesly uživatel
```

Způsob zápisu hesla ovlivníme několika přepínači:

- `-d` — použije se algoritmus CRAPT
- `-m` — použije se algoritmus MD5
- `-p` — heslo je v otevřeném textu, není zašifrováno
- `-s` — použije se algoritmus SHA

Zrušení účtu uživateli je jednoduché

```
# htpasswd -D soubor-s-hesly uživatel
```

43.8. Apache-SSL

Apache-SSL instalujeme stejně jako Apache. V průběhu instalace budeme vyzváni k vyplnění údajů potřebných pro vytvoření SSL certifikátu serveru. Po instalaci rovněž postupujeme v konfiguraci obdobně.

43.9. PHP

* *section id="php" xreflabel="PHP"*

Po nainstalování php4 jsem zjistil že mi nefunguje. V deníku `/var/log/apache-ssl/error.log` se objevila následující chyba:

```
[Fri Mar 11 12:54:52 2005] /usr/lib/apache-ssl/gcache started
[Fri Mar 11 12:54:53 2005] [error] (2)No such file or directory: mod_mime_magic:\
 can't read magic file /etc/apache-ssl/share/magic
```

Toto je spůsobeno přehlédnutím jednoho dialogu kdy se nás instalátor php4 ptal zdali chceme aby za nás opravil konfiguraci Apache a/nebo 43.8. Tím se do konfiguračního souboru nezapsal řádek:

LoadModule php4_module /usr/lib/apache/1.3/libphp4.so

43.10. Apache 2

FIXME:

43.10.1. Apache2 v Debian Sarge

Instalace je jednoduchá.

```
# aptitude install apache2 apache2-doc
```

A můžeme ověřit že je apache nakonfigurovaný. Přímo ze serveru můžem přistoupit na titulní stránku.

```
# links http://localhost
```

Pokud potřebujeme apache se ssl, tedy přistupovat na něj šifrovaně přes https, povolíme modul ssl.

```
# a2enmod ssl
Module ssl installed; run /etc/init.d/apache2 force-reload to enable.
# /etc/init.d/apache2 force-reload
```

Vytvoříme certifikát s platností na 10 let. Doba platnosti certifikátu je zadána jko počet dnů parametrem `-days`.

```
# apache2-ssl-certificate -days 3650

creating selfsigned certificate
⋮
Country Name (2 letter code) [GB]:CZ
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Praha
Organization Name (eg, company; recommended) []:Nasefirma, a.s.
Organizational Unit Name (eg, section) []:IT Departement
server name (eg. ssl.domain.tld; required!!!) []:www.example.com
CertificateFile /etc/CA/server_certs/krecek.giga-net.cz-cert.pe
Email Address []:admin@example.com
```

Pokud potřebujeme vytvořit certifikát znovu, například proto že jsme v něm udělali chybu nebo jsem nezadali správnou dobu platnosti certifikátu, použijeme k tomu přepínač `--force`.

```
# apache2-ssl-certificate --force -days 8000
```

V souboru `/etc/apache2/sites-availabel/default` zaměníme řádky

```
NameVirtualHost *
<VirtualHost *>
```

za

```
NameVirtualHost *:80
<VirtualHost *:80>
```

A do souboru `/etc/apache2/ports.conf` přidáme

```
Listen 443
```

`apache2-ssl-certificate`

```
CertificateFile /etc/CA/server_certs/krecek.giga-net.cz-cert.pem
       SSLCertificateKeyFile /etc/CA/server_certs/krecek.giga-net.cz-key.pem
```

43.10.2. Virtuální weby

FIXME:

Soubor `virtualhost.conf`:

```
##############################################################################
NameVirtualHost 172.20.1.8
##############################################################################

# Globalni nastaveni
<Directory /srv/virtual/s8:
 Options Indexes FollowSymlinks
 AllowOverride None

 Order allow,deny
 Allow from all
</Direcotry>

<VirtualHost 172.20.1.8>
 ServerAdmin ....
 ServerName s8.cz
 Redirect / http://www.talent.cz/talent/author.php3?id_a=104
</VirtualHost>

<VirtualHost 172.20.1.8>
        ServerName www.s8.cz
        DocumentRoot /srv/virtual/s8
</VirtualHost>
```

43.10.3. title

```
# rpm -i libapr0-2.0.49-27.8.i586.rpm
# rpm -i apache2-prefork-2.0.49-27.8.i586.rpm apache2-2.0.49-27.8.i586.rpm
⋮
Reading /etc/sysconfig and updating the system...
Executing /sbin/conf.d/SuSEconfig.apache2...
Looking for multi-processing modules (MPM)
  APACHE_MPM is unset (/etc/sysconfig/apache2), picked "prefork"
  create symbolic link ‘/usr/sbin/httpd2’ to ‘/usr/sbin/httpd2-prefork’
```

```
  create symbolic link ‘/usr/share/apache2/build/config_vars.mk’ to ‘config_vars.mk-prefork’
Module "php4" is not installed, ignoring.
Finished
```

```
/etc/apache2
/var/log/apache2
/srv/www
```

```
/etc/apache2/httpd.conf
```

FIXME:

```
/etc/apache2/
```

FIXME:

```
<Directory />
    Options SymLinksIfOwnerMatch
    AllowOverride None
</Directory>
```

Document Root:

```
<Directory /var/www/>
    Options Indexes Includes FollowSymLinks MultiViews
    AllowOverride None AuthConfig

    Order allow,deny
    Allow from all
</Directory>
```

```
<IfModule mod_userdir.c>
  UserDir public_html
</IfModule>
```

```
<Directory /home/*/public_html>
  AllowOverride
  ⋮
</IfModule>
```

Doporučení: Dát sambovská a www data na zvláštní svazek.

Soubor `.htaccess` řídí přístupová práva do adresáře ve ketrém se nachází.

```
<Files ~ "^\.ht">
  Order allow,deny
  Deny from all
</Files>
```

```
CustomLog /var/log/apache/access.log full
```

Aliasy

```
Alias /icons/ /usr/share/apache/icons/
<Directory /usr/share/apache/icons>
  Options Indexes MultiViews
  AllowOverride None
```

```
    Order allow,deny
    Allow from all
</Directory>

ScriptAlias /cgi-bin/ ...
```

Virtual host

```
NameVirtualHost _default_:80

<VirtualHost 313.74.7.226>
    ServerAdmin ...
    DocumentRoot /home/apache/weby/cvs-domain-cz
    ServerName  cvs.domain.cz

    Alias /doc/ /usr/share/doc/
    <Directory /home/apache/weby/cvs-domain-cz>
        AllowOverrride None AuthConfig
        Options ExecCGI
        AddHandler cgi-script .cgi
    <Directory>
</VirtualHost>
```

43.11. Řízení přístupu ke stránkám

43.11.1. Statické řízení pomocí konfigurace

Pro omezení přístupu můžeme použít „statickou“ konfiguraci apache. Tato může být buď'to v konfiguračních souborech apache v adresáři `/etc/apache2/...`, nebo v souboru `.htaccess` umístněném v adresáři přístup ke kterému chceme řídit (omezovat).

V následující ukázce jsem spojil dvě metody omezení přístupu. První je pomocí IP adresy odkud dotaz na www stránky přichází. V daném případě povolujeme přístup z IP adres 192.0.2.1 a 192.0.2.16 az 192.0.2.31 vyjma adresy 192.0.2.21. Druhá blok příkazů konfuguruje omezení přístupu heslem. V souboru `/etc/apache2/passwd` jsou hashe hesel a příkazem `Require user` říkáme kteří z uživatelů mají k adresáři přístup.

```
<Directory /var/www/karel>
        AllowOverride None

 # Řízení přístupu IP adresou uživatele
        Order allow,deny
        Allow from 192.0.2.1 192.0.2.16/28
 Deny from 192.0.2.21

        # Řízení přístupu heslem
        AuthType        Basic
        AuthName        "Tajne informace"
        AuthUserFile    /etc/apache2/passwd
        Require user    karel ivana
</Directory>
```

Hesla v souboru udržujeme programem **htpasswd**. Při vytváření prvního uživatele soubor vytvoříme.

```
# htpasswd -c /etc/apache2/passwd karel
passwd:
```

Při přidávání dalších uživatelů, nebo při změně hesla stávajících použijeme příkaz:

```
# htpasswd /etc/apache2/passwd ivana
passwd:
```

Abych si oddělil konfigurace, zapasal jsem celý blok příkazů tak jak je uveden výše do souboru `/etc/apache2/conf.d/karel`.

Pokud nemůžeme použít konfigurační soubory apache, například publikujeme na serveru kde nemáme administrátorský přístup a jsme jen běžní uživatelé, použijeme možnost uložit konfiguraci do souboru `.htaccess`. Tento soubor obsahuje výše uvedený blok mimo `<Directory ...` a `AllowOverride`.

Kapitola 44. Apache2

FIXME:

44.1. Instalace a konfigurace

Instalační a konfigurační postupy pro různé OS a jejich verze.

44.1.1. Debian Lenny

Odkazy:

-
-
-

44.2. Základní konfigurace

Konfigurace apache2 je uložena v adresářové struktuře `/etc/apache2`. Zde je hlavní konfigurační soubor `apache2.conf` a další. Moduly do apache které si nainstalujeme mají konfigurace v adresáři `mods-available`. Ty které jsou aktuálně použity, tedy které apache načítá, jsou v adresáři `mods-enabled`. Zapnutí/vypnutí modulu provedeme přikazy:

```
# a2enmod jméno-modulu
# a2dismod jméno-modulu
```

V adresáři `sites-available` jsou pak konfigurace virtuálních webů a v adresáři `sites-enabled` pak ty weby které jsou aktuálně zapnuty. Pro zapnutí/vypnutí virtuálního webu jsou obdobné příkazy jako pro moduly:

```
# a2ensite jméno-webu
# a2dissite jméno-webu
```

44.3. Virtuální weby

Konfigurace virtuálních webů jsou v adresáři `/etc/apache2/sites-available`.

* *Ukázky převzaté z http://www.debianhelp.co.uk/virtualhosts.htm.*

Příklad 44-1. Soubor /etc/apache2/sites-available/exmple

```
<VirtualHost *>
    ServerName www.example.com
    DocumentRoot /home/www/htdocs/example.com
    ServerAdmin webmaster@example.com
    ErrorLog /var/log/apache2/www.example.com-error_log
    CustomLog /var/log/apache2/www.example.com-access_log common
</VirtualHost>
```

Příklad 44-2. Soubor /etc/apache2/sites-available/myothercompany

```
<VirtualHost *>
    ServerName www.myothercompany.com
    DocumentRoot /home/www/htdocs/myothercompany.com
    ServerAdmin webmaster@myothercompany.com
    ErrorLog /var/log/apache2/www.myothercompany.com-error_log
    CustomLog /var/log/apache2/www.myothercompany.com-access_log common
</VirtualHost>
```

* ***FIXME:****Následující konfiguraci nahradit něčím smysluplnějším.*

Příklad 44-3. Ukázka konfigurace virtuánlího webu rt

```
NameVirtualHost rt:443
<VirtualHost rt:443>
        ServerName      rt.example.com

 # Konfigurace SSL
 SSLEngine On
 SSLCertificateFile /etc/apache2/ssl/crt/rt.crt
 SSLCertificetaKeyFile /etc/apache2/ssl/key/rt.key

 Include /etc/request-tracker3.6/apache2-modperl2.conf
</VirtualHost>
```

Příklad 44-4.

```
<VirtualHost *:80>
        ServerAdmin admin@example.com
        ServerName intranet.example.com
        DocumentRoot /usr/local/share/intranet-1.23

        <Directory />
        Options FollowSymLinks
               AllowOverride None
        </Directory>

        <Directory /usr/local/share/intranet-1.23>
                Allow from 10.0.0.0/8 192.168.1.0/8
                Allow from 23.59.312.97
 </Directory>
<VirtualHost>
```

44.4. Virtuální weby založené na jménu

* *Příklad pochází z http://httpd.apache.org/docs/2.0/vhosts/name-based.html.*

```
 NameVirtualHost *:80

<VirtualHost *:80>
ServerName www.domain.tld
ServerAlias domain.tld *.domain.tld
DocumentRoot /www/domain
```

```
</VirtualHost>

<VirtualHost *:80>
ServerName www.otherdomain.tld
DocumentRoot /www/otherdomain
</VirtualHost>
```

Direktiva `NameVirtualHost` je použita jen jednou a parametry zadávané u `VirtualHost` jsou textově identické jak ten u `NameVirtualHost`.

Poznámka: Podrobnější popis toho jak virtuální weby v Apachi fungují lze nalézt na http://httpd.apache.org/docs/2.0/vhosts/details.html.

44.5. SSL

Je třeba nechat obsluhovat Apache2 i https port číslo 443. To učiníme přidáním následujícího řádku do souboru `ports.conf`.

```
Listen 443
```

* *Následující text jsou výpisky z http://snow.nl/dist/htmlc/ch08s02.html*

```
$ openssl genrsa -des3 -out server.key 1024
$ openssl req -new -key server.key -out server.csr
```

Soubor `server.csr` je žádost o podpis certifikátu, kterou si necháváme podepsat certifikační autoritou.

V konfiguraci apache se pak na příslušném místě nachází direktivy:

```
SSLCertificateFile    /path/to/this/server.crt
SSLCertificateKeyFile /path/to/this/server.key
```

44.5.1. Vytvoření certifikátu

Odkazy:

- Creating a Self-signed SSL Certificate for Apache 2 on Debian lenny (http://blog.wearesakuzaku.com/creating-a-self-signed-ssl-certificate-for-apache-2-on-debian-lenny/)
- Debian, Apache2 + ssl + mod_rewrite (http://www.jens.cz/debian-apache2-ssl-mod_rewrite/)
- Debian Lenny - Apache, SSL and Virtual Hosts (http://cloudservers.mosso.com/index.php/Debian_Lenny_-_Apache,_SSL_and_Virtual_Hosts)
-
-
-

```
# openssl req -new -x509 -days 3772 -nodes -out /etc/apache2/ssl/apache-cert.pem -keyout /etc/
...
# chmod 600 apache-key.pem
```

Počet dní platnosti certifikátu získáme výpočtem třeba v Ruby.

```
$ irb
irb(main):002:0> require 'date'
=> true
irb(main):014:0> dny=Date::parse('2019-12-30')-Date::today()
=> Rational(3772,1)
```

44.6. Virtuální web hosting a SSL

V originální dokumentaci k apache se vyskytuje věta

> Name-based virtual hosting cannot be used with SSL secure servers because of the nature of the SSL protocol.

Pokus vyheldat další informace na netu mě přivedl na následující stránky, které se k možnosti virtuálního SSL hostingu vyjadřují podrobněji.

- http://mail-archives.apache.org/mod_mbox/httpd-docs/200307.mbox/%3C33352.192.168.2.254.1058352330.squirrel@web
- http://httpd.apache.org/docs/2.0/ssl/ssl_faq.html#vhosts2
- http://httpd.apache.org/docs-2.0/ssl/ssl_faq.html#vhosts
- Setting up an SSL VirtualHost on a NameVirtualHost IP address (https://wiki.hss.caltech.edu/help/SetupSslNameVirtualHost)
- http://mailman1.u.washington.edu/pipermail/pubcookie-users/2004-July/000551.html
-

44.7. Řízení přístupu k stránkám prostředky Apache

Odkazy:

- `.htaccess` files (http://httpd.apache.org/docs/1.3/howto/htaccess.html)

U Apache máme možnost ovlivnit konfiguraci lokálně pro jednotlivé adresáře.

Kapitola 45. SQL server

Šablona pro nové kapitoly

* *chapter id="sql"*

Tato kapitola pojednává o SQL serverech a to z hlediska administrátor. V linuxu je k dispozici řada SQL serverů, tady popíši jen některé z nich. Jsou to zejména dva dle mého názoru nejrozšířenější a to MySQL a PostgreSQL.

45.1. MySQL

* *section id="mysql" xreflabel="MySQL"*

Odkazy:

- MySQL manuál (http://mm.gene.cz/)

Popis instalace, konfigurace a užití databáze MySQL.

45.1.1. Instalace a konfigurace

* *section id="mysql.instalace" xreflabel="Instalace"*
* ***FIXME:****popsat instalaci*

```
# apt-get install mysql-server
```

Odpovíme instalátoru na několik otázek. Jestli chceme v případě odinstalace odstranit i databáze. Na tuto otázku je lépe odpovědět „ne“ abychom v takovém případě nepřišli o databáze. A poslední otázkou je má-li se startovat mysql server automaticky při startu počítače či nokoliv.

Po úspěšné instalaci, předtím než MySQL spřístupníme uživatelům ji musíme zabezpečit. Zabezpečení sestává z

- nastavení hesla správci databáze (`root`)
- odstraníme anonimní oprávnění přístupu
- nastavíme přístupové metody k databázi

Počáteční nastavení hesla správce databáze, uživatele `root` je důležité, neb databáze po instalaci nemá správcovský účet zaheslován. Použijeme tedy například příkaz

```
# mysqladmin -u root password "heslo"
```

Po nastavení hesla uživateli `root` zrušíme anonymní přístup

```
# mysql -p -u root mysql
Enter password: heslo
mysql> DELETE FROM user WHERE User=”;
mysql> DELETE FROM db WHERE User=”;
mysql> FLUSH PRIVILEGES;
mysql> \q
```

45.1.1.1. Výběr konfigurace

Pokud nám nevyhovuje standardní konfigurace, se kterou se MySQL nainstaluje, můžeme využít některého z dodaných konfiguračních souborů. Tyto jsou v adresáři `/usr/share/doc/mysql-server-5.0/examples`. Podle popisu v soubory si vybereme ten správný a umístníme ho místo `/etc/mysql/my.cnf`. Zkontrolujeme, že na konci souboru je:

```
#
# * IMPORTANT: Additional settings that can override those from this file!
#
!includedir /etc/mysql/conf.d/
```

To proto aby se braly v úvahu naše úpravy konfigurace v adresáři `/etc/mysql/conf.d/`.

45.1.1.2. Deníky

Implicitní instalace nevytváří žádně deníky mimo binární žurnál. Jen přes **syslogd** do souboru `/var/log/syslog` jdou nejzákladnější informace o startu a ukončování MySQL. Vytváření deníků zapneme přidáním nebo odkomentováním řádků v konfiguraci v souboru `/etc/mysql/my.cnf`. Já využiji jiné možnosti. Na konci souboru my.snf je příkaz `!includedir /etc/mysql/conf.d/`. Tento načte kousky konfigurací z adresáře `/etc/mysql/conf.d`. A právě do tohoto adresáře jsem umístnil soubor `log.conf` s obsahem:

```
[mysqld]
# * Logging
log_slow_queries = /var/log/mysql/mysql-slow.log
long_query_time = 2
log_warnings
#log = /var/log/mysql/mysql.log
```

45.1.2. Administrace

Odkazy:

- Simple MySQL cookbook (http://www.debian-administration.org/article/Simple_MySQL_cookbook)

Následující text je zatím jen sbírkou poznámek které jsem si zapsal při čtení knih.

Vypsání řady proměnných jenž popisují vlastnosti a nastavení serveru. Z příkazové řádky můžeme použít:

```
# mysqladmin variables
```

Z SQL konzoly pak příkazem:

```
mysql> SHOW VARIABLES
```

Běží-li na hostu více sql serverů, používá každý jiný port. Při dotazování serveru pak musíme říci který máme na mysli.

```
# mysqladmin --host=localhost --port=číslo_portu variables
```

Je-li server přístupný přes soket, příkaz bude vypadat takto:

```
# mysqladmin --host=localhost --socket=/cesta/k/soketu/ variables
```

Jak je patrno server je přístupný více způsoby (komunikačními kanály) a to současně. Jsou to:

- UNIX domain socket — na UNIXu
- TCP/IP port — na UNIXu a WinNT
- named pipe — na WinNT

K stávající databáz se přípojíme příkazem **mysql** tento nám otevře spojení a spřístupní CLI databáze. Následující řádky ukazují příklady takových připojení.

```
$ mysql -h localhost -u root
$ mysql -h cobra.snake.net -u root
```

45.1.2.1. Struktura adresáře s daty

Adresář s daty je určen proměnnou `datadir` jejíž hodnotu získáme třeba příkazem

```
mysql> SHOW VARIABLES LIKE 'datadir';
```

Adresář obsahuje podadresář pro každou databázi. Jednotlivé tabulky v databázi jsou reprezentovány několika soubory v podadresáři databáze. Výjimkou je InnoDB.

> **Poznámka:** Pokud budete programy **myisamchk** a **isamchk** opravovat soubory s daty musí být SQL server(y) zastaveny. Opravu také může provádět server příkazy `CHECK TABLE` a **REPAIR TABLE**

Zpět tedy k databázím. Každá databáze je ve vlastní složce jenž se jmenuje stejně jako tato databáze. Je tedy dostupná jako `DATADIR`/*`jméno_databáze`*.

45.1.2.2. Vytváření a administrace databází

Odkazy:
- CREATE DATABASE Syntax (http://dev.mysql.com/doc/refman/5.0/en/create-database.html)
-

Seznam databází získáme příkazem

```
mysql> SHOW DATABASES
```

Vytvoření databáze dosáhneme příkazem

```
mysql> CREATE DATABASE [IF NOT EXIST] název_databáze [[DEFAULT] CHARACTER SET charset_name] [,[DE
```

Odstranění/Zrušení databáze pak provedeme jednoduše příkazem

```
mysql> DROP DATABASE název_databáze
```

Ukázka vytvoření databáze `templar`

```
mysql> CREATE DATABASE tomb;
Query OK, 1 row affected (0.19 sec)
```

Povolení přístupu k databázi

```
mysql> GRANT ALL PRIVILEGES ON tomb.* TO tomas;
Query OK, 0 rows affected (0.00 sec)
```

Ukázka vytvoření databáze.

```
mysql> CREATE DATABASE ceniky CHARACTER SET 'utf8';
```

45.1.2.3. Účty a jejich správa

Přiřazení hesla účtu

```
$ mysqladmin -h localhost -u root password "heslo_uživatele"
$ mysqladmin -h cobra.snake.net -u root password "heslo_uživatele"
```

Tyto dva příkazy nastaví přístupové heslo uživateli `root` pro přístup z lokálního stroje (`localhost`) a přes síť ze stroje `cobra.snake.net`. Stejného efekru dosáhneme z CLI takto

```
$ mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('heslo_uživatele');
mysql> SET PASSWORD FOR 'root'@'cobra.snake.net' = PASSWORD('heslo_uživatele');
```

Tabulku s hesly můžeme upravit přímo

```
$ mysql -u root
mysql> use mysql;
mysql> UPDATE user SET Password=PASSWORD('heslo_uživatele') WHERE User='root';
mysql> FLUSH PRIVILEGES;
```

Při přihlašování k účtu který je chráněn heslem musíme na toto **mysql** upozornit a ten se nás na heslo zeptá.

```
$ mysql -p -u root
Enter password: heslo_uživatele
mysql>
```

Příklad 45-1. Příklady z dokumentace MySQL

```
myslq> CREATE USER 'monty'@'localhost' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'@'localhost' WITH GRANT OPTION;
mysql> CREATE USER 'monty'@'%' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'@'%' WITH GRANT OPTION;
mysql> CREATE USER 'admin'@'localhost';
mysql> GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost';
mysql> CREATE USER 'dummy'@'localhost';
```

Pokud máme staršího klienta a zkoušíme se připojit k novějšímu server, může nastat chyba:

ERROR 1251: Client does not support authentication protocol requested by server; consider upgrading MySQL client

Tento případ se dá vyřešit použitíme staršího způsobu hashování hesel, funkce `OLD_PASSWORD()`.

```
mysql> SET PASSWORD FOR 'uzivate'@'server' = OLD_PASSWORD('heslo');
```

* *http://lists.mysql.com/mysql/183477*

45.1.2.4. Vytváření nových uživatelů a udělování práv

Pro vytváření účtů a přidělování práv slouží příkaz **GRANT**. Jeho úplná syntaxe je:

GRANT *oprávnění* (*sloupce*)
 ON *pro_co*

TO *účet* IDENTIFIED BY '*heslo*'
REQUIRE *požadavky_na_šifrování*
WITH *volby_pro_udělování_práv*

Uživatelé a jejich účty jsou popsány v tabulce `user` v databázi `mysql`. Veškerá administrace uživatelů je jen manipulace s touto tabulkou. Pro jednoduchost si uvedeme ukázky některých základních administrativních úkonů.

Vytvoření nového uživatele:

```
mysql> use mysql
mysql> INSER INTO user (user,host) VALUES ('tomas', 'station.firma.cz');
Query OK, 1 row affected (0.02 sec)

mysql> GRANT ALL ON tomb.* TO tomas IDENTIFIED BY 'heslo123';
```

Odstranění/zrušení existujícího uživatele provedeme smazáním příslušné řádky v tabulce. Protože uživatelé jsou identifikováni jménem a stroje ze kterého přistupují nesmíme zapomenout do klauzule **WHERE** uvést omezující podmínku na sloupec `host`. V opačném případě smažeme všechny záznamy stejného jména, což nemusí být vždy žádoucí.

```
mysql> DELETE FROM user WHERE user='tomas' AND host='station.firma.cz');
Query OK, 1 row affected (0.15 sec)
```

Pro změnu hesla uživatel můžeme použít příkaz **SET PASSWORD** například takto

```
mysql> SET PASSWORD FOR 'tomas'@'station.firma.cz' = PASSWORD('heslo123');
ERROR 1133: Can't find any matching row in the user table
```

Nebo můžeme standardním **UPDATE** přímo změnit odovídající řadek či řádky v tabulce `user`

```
mysql> UPDATE user SET password=PASSWORD('heslo123') WHERE user='tomas' AND host='station.firm
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> CREATE DATABASE mojedb;
mysql> CREATE USER ja;
mysql> GRANT ALL PRIVILEGES ON mojedb.* TO 'ja'@'%' IDENTIFIED BY 'heslo' WITH GRANT OPTION;
```

45.1.2.5. Oprava poškozených tabulek

```
mysql> REPAIR TABLE tabulka;
```

45.1.3. Zálohování

MySQL se zálohuje tak jako ostatní SQL databáze dumpem. Záloha všech databází do souboru `backup.sql` se provedé příkazem **mysqldump** s přepínačem `--all-databases`:

```
# mysqldump --all-databases --opt >backup.sql
```

Budeme-li chtít zálohovat jen jednu databázi, například `projekt`, uvedeme ji jako parametr.

```
# mysqldump --opt projekt >backup.sql
```

Hotovou zálohu uschováme třeba na pásce nebo na jiném médiu. Potřebujeme-li databázi obnovit, obnovíme si z pásky dump databáze `backup.sql`, databázi vyčistíme a inicializujeme ji připraveným souborem.

```
# mysql < backup.sql
```

FIXME: Dalším způsobem jak zálohovat je použít program **mysqlhotcopy**. Tent je schopen zálohovat běžící databázi.

```
# mysqlhotcopy sampdb
```

45.1.4. Obnova dat

Data obnovíme pomocí

```
$ mysql -u dbuser db -p <db.zaloha.sql
```

45.1.5. Změna zapomenutého hesla administrátora

* *Podle http://www.debian-administration.org/articles/442**FIXME:***

```
# /etc/init.d/mysql stop
# /usr/bin/mysqld_safe --skip-grant-tables &

$ mysql --user=root mysql
Enter password:

mysql> update user set Password=PASSWORD('nové-heslo') WHERE User='root';
mysql> flush privileges;
mysql> exit
```

Nyní je třeba zastavit server. Nejdříve si jej jako úlohu běžící v pozadí fáme do popředí

```
# fg
```

Pomocí **Ctrl**+**C** jej zastavíme a poté spustíme.

```
# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for corrupt, not cleanly closed and upgrade needing tables..
```

45.2. PostgreSQL

* *section id="postgresql" xreflabel="PostgreSQL"*

Databázový systém postgres je starší než MySQL a je distribuován pod jinou licencí, BSD.

45.2.1. Instalace

FIXME:popsat instalaci

```
Enter default wncoding (SQL_ASCII): UNICODE   (LATIN2)
Select Locale [C]: cs_CZ         (cs_CZ/en_US)
```

45.2.1.1. Instalace v Debian Lenny

Standardní verze postgresu v Debian Lenny je verze 8.3. Pokud použijeme běžný instalační postup, nainstaluje se ta.

```
# aptitude install postgresql
```

Od verze Etch jsou konfigurační i datové adresáře udělány tak, aby byl možný současný provoz nejen více clusterů ale i různé verze Postgresu. V standardní distribuce je ovšem jen verze 8.3.

Konfigurační soubory pro každý cluster jsou v samostatném adresáři `/etc/postgresql/`*`verze/cluster`*. Standardně jsou tedy v adresáři `/etc/postgresql/8.3/main`.

45.2.1.2. Instalace v Debian Etch

V Debian Etch jsou k dispozici dvě věrze postgresu. Standardně se instaluje verze 7.4 a můžeme nainstalovat taky novější verzi 8.1. V dalších postupech uvažuji jen standardně instalovanou verzi 7.4.

FIXME:TBD.

```
# aptitude install postgresql
```

Po nainstalování se můžeme skusně připojit k serveru.

```
# su - postgres -c "psql template1"
⋮
template1=# \l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+----------
 template0 | postgres | UNICODE
 template1 | postgres | UNICODE
(2 rows)

template1=#
```

45.2.1.3. Instalace v Debian Sarge

* *section id="postgresql.install.sarge" xreflabel="Instalace v Debian Sarge"*

FIXME:TBD

```
# aptitude install postgresql
```

- What locale should be used by the database backed?: C
- Choose European od US day/month order in dates.: European

```
# aptitude install postgresql-client
```

Ověříme si že máme přístup do databáze.

```
# su postgres psql template1
Welcome to psql 7.4.7, the PostgreSQL interactive terminal.
⋮
template1=#\l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+-----------
 template0 | postgres | SQL_ASCII
 template1 | postgres | SQL_ASCII
(2 rows)
```

45.2.2. Úlohy

45.2.2.1. Vytvoření databáze

Novou databázi můžeme vytvořit dvěma různými způsoby. Prvním z nich je použití programu **createdb**.

```
# su - postgres
$ createdb mojedb
```

```
=# CREATE DATABASE mojedb WITH OWNER=já TEMPLATE=template0 ENCODING='utf-8';
```

45.2.2.2. Zpřístupnění databáze uživateli

Pro zpřístupnění uživateli použijeme program `/usr/bin/createuser`.

45.2.3. Zálohování a obnova

Jednou z možností je vytvořit dump databáze ve formě SQL příkazů. Takovýto dump se načítá tak, jako bychom zadávali příkazy z konzoly. Jeho výhodou je že s pomocí běžných textových nástrojů jej můžeme upravit podle potřeby.

```
pg_dump -Fp db >db.sql
```

```
psql -f db.sql template1
```

45.2.4. Clustery a ruzne verze PostgreSQL

Odkazy:

- postgresql clustering and Debian (http://www.progsoc.org/~wildfire/notes/postgresql-cluster.html)

45.3. Firebird

45.3.1. Instalace Firebird 1.5 do Debian Lenny

Odkazy:

- Debian Lenny Firebird 1.5 (http://www.opennet.ru/openforum/vsluhforumID1/84254.html)
-

Nejdříve opravíme `/etc/apt/sources.list`. Přidáme do něj řádku připojující repositář předchozí verze Etch.

```
deb http://ftp.cz.debian.org/debian/ etch main
```

Nyní můžeme instalovat.

```
# aptitude install -t etch firebird2-super-server
```

Doinstalujeme nástroje.

```
# aptitude install -t etch firebird2-utils-super
```

45.3.2. Firebird 2.1

Náš nový produkční software beží na serverové části na SQL serveru FireBird. Bohužel dodavatelská firma nepodporuje v této chvíli stabilní verzi 2.0.x jenž je jak v Debian Etch tak v Debian Lenny. Vyžadují přinejmenším verzi 2.1 nebo 2.5 která je označena jako beta i vývojáři. Jak takovou situaci řešit?

Odkazy:

- building firebird 2.1 from debian git repository (http://firebirdnews.blogspot.com/2008/02/building-firebird-21-from-debian-git.html)

Při hledání na Internetu jsem narazil na článek building firebird 2.1 from debian git repository (http://firebirdnews.blogspot.com/2008/02/building-firebird-21-from-debian-git.html). Na pracovním notebooku s instalací Lenny na architektuře amd64 (x86-64) jsem použil postup

```
# aptitude install devscripts
# aptitude install docbook-to-man bison libicu-dev libedit-dev autoconf libtool automake quilt
$ mkdir -p ~/work/fb
$ cd ~/work/fb
$ git-clone git://git.debian.org/git/pkg-firebird/2.1
$ cd 2.1
$ debuild -i
```

45.3.3. Firebird 2.5

* *Postponed*

```
wimp:/var/packages# LANG=C DEBIAN_FIREBIRD_DEBUG="yes" dpkg --configure  firebird2.5-server-comm
```

45.3.4. Záloha a obnova databáze

Záloha se obnova se provádí příkazem **gbak**.

```
# gbak
```

Kapitola 46. Jednoduché zpracování textu

Šablona pro nové kapitoly

Jednoduché zpracování textu, nebo spíše zpracování jednoduchého textu.

UNIX už ve svých začátcích byl orientován na pořizování a zpracování textů. Má pro tento účel řadu programů.

Nejdříve si povíme co je to jednoduchý text. Jedná se o prostý text v kódování ASCII či některém 8-mi bitovém rozšíření, například ISO-8859-2, ve kterém můžeme prezentovat i národní znaky. Text je psán do řádků a neobsahuje žádné „binární“ informace. Ukázka:

```
Toto je prostý text jenž se dá
jednoduše zpracovávat.
```

Na textu se dají vybírat jeho části, provádět jednoduché transformace a editační operace, zjišťovat velikost. Jednotlivé skupiny programů ve skratce:

Zjišťování informací o textu

- **file** — zjistí jakého druhu/typu soubor je.

Výber části z textu

- **grep**, **egrep**, **fgrep** —
- **head** —
- **tail** —
- **cut** —

Editace, změny a filtrování

- **tr** —
- **sed** —
- **col** —
- **sort** —
- **expand** — convert tabs to spaces

Prostředky pro komplexnější změny

- **awk** —
- **perl** —

Kapitola 47. Postscript

Nástroje pro manipulaci s postrscirptovými soubory

Seznam balíčků:
- psutils 1.17-13 sbírka nástrojů pro práci s postscriptem
- psptools 1.2.2-5 nástroje pro postscriptové tiskárny a zařízení

Programy (nástroje):
- showchar
- psnup
- pstops zpřehází stránky v souboru

47.1. Drobné úlohy

47.1.1. Vsunutí prázdné strany do postscriptového souboru

Tento problém by mohl vyřešit program pstops. Ale neřeší. Nakonec pomohlo pomocí textového editoru najít příslušnou stránku podle

```
%% Page
```

A vložit před ni

```
showpage
```

47.1.2. Posunutí stránek na zrcadle

```
$ cat cvsbook.ps| pstops "2:0(0,0),1(-0.6cm,0)"

$ cat work/book.ps |pstops "2:0(0,2cm),1(-0.6cm,2cm)" \
  |psselect -p1-20 |~/bin/foldprn -s 20
```

47.1.3. Instrukce pro převedení knihy *Common Lisp Book* do Postscriptu

http://pragmaticprogrammer.com/cgi-local/pragprog?Instru...

```
$ wget --recursive --level=4 http://psg.com/~dlamkins/sl/cover.html
$ cd psg.com/*dlamkins/sl/
$ mkdir output
$ c -ax gifs *.html output
$ for i in chapter*.html; do
>     sed -e '/<div/,/<.address/ s/^.*$//' $i >output/$i
> done
$ cd output
```

```
$ html2ps --twoup --number -o sl.ps --xref (author,about,dedic,cred,ack,fore,intr,cont,track,c
$ rm -r *.html gif

# optionally
$ ps2pdf sl.{ps,pdf}
```

Kapitola 48. Zpracování zvuku

48.1. Jednoduché úkony

48.1.1. Konverze RAM to WAV a MP3

Odkazy:

- Convert Realaudio stream to wav / mp3 on linux (http://www.larsen-b.com/Article/127.html)

```
$ mplayer -cache 10 -ao pcm -aofile stream.wav  rtsp://XXXX/stream.rm
$ lame stream.wav
```

VI. Síťové služby orientované na uživatele

Konfigurace a správa služeb

Síťové služby orientované na uživatele.

Poznámka: Rozdělění síťových služeb na uživatelské a služby pro síť je velmi umělé, ne dobře definované.

* *Promyslet přerozdělení kapitol mezi částmi [xref linkend="part.provoz-site"/] a touto částí, připadnš sjednocení do jedné části připadně přerozdělení do úplně jiných částí.*

Kapitola 49. Systémy sledování dotazů, hotline

Šablona pro nové kapitoly

* *chapter id="request_tracker_systems"*

Jedním ze zajímavých druhů programů, zejména pro nasazení ve větší organizaci je program pro sledování žádostí, tzv. *Request Tracker System*. Základním principem programů této třídy je usnadňovat sledování toku žádostí a jejich rozpracovanosti. Ideální nasazení je IT odděleních větších či velikých firem a v technické podpoře zákazníků.

Odkazy a zdroje:

- Request Tracker (http://www.bestpractical.com/rt/)
- Double Choco Latte (http://dcl.sourceforge.net)
- Scarab
- Tech Tracker (http://lee.k12.nc.us/~joden/lcss/tech_tracker/) is a web-based IT tracking system. The goal of the project is to provide a simple to administrate and use, yet powerful IT tracking system. The interface should be usable even over slow connections and with any web client (works with Lynx).
- Tutos (http://www.tutos.org)
- phpGroupWare (http://www.phpgroupware.org) - formerly known as webdistro - is a multi-user groupware suite written in PHP.

  It provides about 50 web-based applications, as there are the Calendar, Addressbook, an advanced Projects manager, Todo List, Notes, Email, Newsgroup- and Headlines Reader, a Filemanager and many more Applications. The calendar supports repeating events and includes alarm functions. The email system supports inline graphics and file attachments.

  The system as a whole supports user preferences, themes, user permissions, multi-language support and unser groups. It includes modules to setup and administrate the working environment. The groupware suite is based on an advanced Application Programming Interface (API).

Systémy lístků, hotline, sledování dotazů, ... jsou v zásadě realizují celý životní krok dotazu. Jejich primární použití je ve firmách na které se obrací větší množství zákazníků. Tyto dotazy je třeba sledovat, zajistit že se neztratí a budou ke spokojenosti zákazníka vyřešeny v co nejkratším čase. Dají se i nasadit v pracovních skupinách kde slouží jako společný seznam úkolů jenž je třeba sledovat.

Já jsem se začal tímto druhem programů zabývat v době kdy jsem potřeboval vybrat nástroj pro sledování stavu úkolů vlastních a kolegů v oddělení IT.

49.1. Request Tracker

* *section id="rt"*

Prvním systémem který mne zaujal natolik že jsem se jej rozhodl vyzkoušet byl Request Tracker (http://www.bestpractical.com/rt/) dále jen RT. Důvod proč mne zaujal byla jeho koncentrace jen na jeden problém, sledování žádostí. RT neřeší jiné problémy než zde uvedený. Podle popisu na webu vypadal jako dostatečně silné a jednoduché řešení. Nechtěl jsem ve firmě začít zkoušet nic moc složitého, dokud se lidi nenaučí pracovat s jednoduchým nástrojem který jim stačí. V čemkoliv složitějším by mohli ztrácet orientaci a utrácet svou energii a drahocenný čas zápasením z příliš složitým systémem. Až se naučí do detailů využívat jednoduchý systém, mohu přijít s nabídkou systému složitějšího jako s alternativou. Abyste získali přehled, k čemu všemu nám může RT posloužit, připravil jsem několik vzorových/modelových situaích jenž sou popsány na konci v části o použití RT. Ale dost už obecného povídání a věnujme se instalaci.

49.1.1. Instalace

* *section id="rt.instalace"*
* *rcsinfo="$Id: ch-request_systems.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $"*

V rámci instalace a konfigurace se nazabývám nasazení RT na obecný UNIX. Maximálně využívám výhod Debian GNU/Linux a jaho balíčkovacího systému. Jediným problémem byl požadavek nasadit ne zrovna nejstarší verzi programového vybavení. Vzhledem k tomu že ve Woody (aktuální stabilní verze) je RT ve verzi 1.0.7-2 jako balíček `request-tracker1`, a na Debian Backports (Backports.ORG) (http://www.backports.org/) tento software není, sáhl jsem do Sarge (aktuální testing) kde je ve verzi 3.0.12-6. Protože celý Request Tracker provozuji ve virtuálním serveru, nebyl pro mne až takový problém jej celý nainstalovat/upgradovat na Sarge.

Předtím než přikročíme k samotné instalaci Request Trackeru, musíme si připravit potřebné prostředí. RT jako webová aplikace potřebuje ke své činnosti webový server. Pro začátek jsem se rozhodl použí Apache s mod-perl.

```
rt:~# apt-get install apache
```

Apache nakonfigurujeme tak, aby fungoval. Protože používám virtuální server, musím nastavit minimálne adresu a port na kterém bude Apache poslouchat.

```
Listen  10.16.66.130:80
BindAddress 10.16.66.130
Port 80
```

Nezapomenu taky nastavit jméno serveru.

```
ServerName rt.firma.cz
```

Poté co si ověříme že apache funguje, a vidíme jeho standardní titulní stránku, přikročíme k samotné instalaci RT. Ještě předtím však zmíním, že ve firmě provozuji databázový server PostgreSQL a rozhodl jsem se jej pro účely RT využít jako databázového *backendu*.

```
rt:~# apt-get install request-tracker3 postgresql-client
```

Klintskému PostgreSQL nastavíme jméno našeho databázového serveru (`/etc/postgresql/postgresql.env`)

```
PGHOST=sql.firma.cz
```

Dále je třeba povolit z rt.firma.cz přístup do SQL databáze, vytvořit uživatele a vytvořit samotnou databázi (tabulky). Do souboru `/etc/postgresql/pg_hba.conf` přidáme povolení přístupu z RT serveru. Přístup bude chráněný heslem.

```
# Request-Tracker 3.0 on virtual server rt.firma.cz
host    rt              10.16.66.130    255.255.255.255         password
```

FIXME:protože vytváření databáze jsem dělal již dávno, nepamatuji si přesný postup. Dopíši jej až budu dělat nějakou jinou instalaci RT.

Nyní nakonfigurujeme RT. Konfigurace kterou budeme měnit je v souboru `/etc/request-tracker3/RT_SiteConfig.pm`

```
Set($rtname, 'rt.firma.cz');
Set($Organization, 'firma.cz');
Set($CorrespondAddress , 'rt@firma.cz');
Set($CommentAddress , 'rt-comment@firma.cz');
Set($Timezone , 'Europe/Prague'); # obviously choose what suits you
```

```
# THE DATABASE:
Set($DatabaseType, 'Pg'); # e.g. Pg or mysql
Set($DatabaseHost   , 'sunrise');
Set($DatabaseRTHost , 'sunrise');
Set($DatabasePort , '5432');
Set($DatabaseUser , 'rt3user');
Set($DatabasePassword , '********');
Set($DatabaseName , 'rt3');

# THE WEBSERVER:
Set($WebPath , "/rt");
Set($WebBaseURL , "http://rt.firma.cz:80");

Set($WebExternalAuth, 1);
Set($WebFallbackToInternalAuth, 1);

1;
```

Ještě nastavíme Apache tak aby správně reagoval na url RT. Do souboru `/etc/apache/httpd.conf` přidáme

```
### Request Tracker
<Location /rt>
        Include         "/etc/request-tracker3/apache-modperl.conf"
</Location>
```

Tím máme v základě RT nakonfigurován a můžeme jej začít používat.

49.1.1.1. Instalace RT3.4 v Debian Sarge

Předpokládám, že již máme někde zprovozněn databázový SQL server. Například podle Instalace Postgres na Debian Sarge.

```
# aptitude install request-tracker3.4
# aptitude install libapache2-mod-perl2

# a2enmod rewrite
```

Vytvoření databáze

```
# su postgres
$ psql template1
Welcome to psql 7.4.7, the PostgreSQL interactive terminal.
⋮
template1=# CREATE USER rtuser WITH PASSWORD 'wibble' CREATEDB NOCREATEUSER;
CREATE USER
```

49.1.2. Instalace na samostatný server

Pokud potřebujeme provozovat jen Request Tracker a žádný jiný software, můžem pro něj použít dedikovaný server nebo WMvare. V takovém případě nesmíme zapomenout na:

- SQL server.
- NTP server pro synchronizaci času.

- Mail server pro mailovou bránu.
- Apache web server.

49.1.3. Instalace do VMware

Do čisté holé instalace Debian Sarge

```
# aptitude install postgresql
# aptitude install apache2 libapache2-mod-perl
# aptitude install request-tracker3.4
# cd /etc/apache2/sites-available
# ln -s /etc/request-tracker3.4/apache2-modperl2.conf request-tracker
# cd
# a2ensite request-tracker
# a2enmod rewrite
# a2enmod actions
# /etc/init.d/apache2 restart
```

Neuspěl jsem s apache2, tak zkouším apache:

```
# aptitude install apache
# cd /etc/apache/conf.d
# ln -s /etc/request-tracker3.4/apache-modperl.conf request-tracker
#
#
#
#
#
#
#
#
#
```

```
rtdb=# CREATE USER root WITH CREATEDB CREATEUSER;
CREATE USER
rtdb=# \q
# rt-setup-database --action schema --datadir /etc/request-tracker3.4/upgrade/3.1.0
Creating database schema.
```

49.1.4. LDAP

Protože udržovat další hesla je velmi nepříjemné, potřeboval jsem přimět RT aby používal hesla z firemního LDAP serveru. Toto nastavíme na dvou místech. První je Apache který převezme starost o ověřování uživatele. Do apache budeme potřebovat modul jenž nainstlaujeme z balíčku.

```
rt3:~# apt-get install libapache-auth-ldap
```

Do konfiguračního souboru `/etc/apache/httpd.conf` pak zapíšeme řádku.

```
LoadModule   auth_ldap_module    libexec/auth_ldap.so
```

Nebo u novější verze jen specifikujeme v době konfigurace, že se má zavést modul `auth_ldap_module`. Konfigurační skript pak zapíše uvedenou řádku do souboru `/etc/apache/modules.conf`.

V původním konfiguračním souboru `/etc/apache/httpd.conf` pak opravíme část specifikující RT takto.

```
### Request Tracker
<Location /rt>
        AuthLDAPURL     ldap://ldap/ou=people,dc=firma,dc=cz?uid
        require         valid-user
        AuthName        "LDAP authenticaton"
        AuthType        Basic
        Include         "/etc/request-tracker3/apache-modperl.conf"
</Location>
```

Apache se tedy bude ptát LDAPu na serveru `ldap.firma.cz` a bude hledat `uid`. Další kontrolu na uživatele neprovádím. Vycházím z toho, že Apache ověří zda uživatel je skutečně ten za koho se vydává a jméno uživatele pak předá RT. To zajistí následující řádky v souboru `/etc/request-tracker3/RT_SiteConfig.pm`.

```
Set($WebExternalAuth, 1);
Set($WebFallbackToInternalAuth, 1);
```

První řádek oznamuje RT že se má spolehnout na externí autentikaci kterou bude provádět Apache a předá jméno uživatele v proměnné. Druhý řádek je pro případ nefunkčního LDAP serveru. I když, nefunguje-li LDAP server, nepustí Apache nikoho k RT.

49.1.5. SQL databáze

Jako SQL databázi používám firemní SQL server jenž je provozován pod PostgreSQL ve verzi 7.2.1-2woody8 který beží na serveru s aliasem `sql`. Na sql serveru jsem do souboru `/etc/postgresql/pg_hba.conf` přidal část

```
# Request-Tracker 3.0 on virtual server rt3.firma.cz
host    rt3             10.16.66.130    255.255.255.255         password
host    template1       10.16.66.130    255.255.255.255         password
```

První řádek povoluje přístup k databázi rt3 řízený heslem. Kde ip adresa `10.16.66.130` je adresou serveru `rt.firma.cz`. Druhý řádek pak povoluje přístup ke ze stejného serveru (`rt.firma.cz`) k databázi template1 jenž je používána v průběhu zakládání a vytváření databáze rt3. Po vytvoření jej můžeme vypustit.

49.1.5.1. PostgreSQL

Zálohování databáze provádím na SQL serveru pravidelně každou noc. Používám k tomu jednoduchý skript.

```
#!/bin/sh
# Create a daily backup of rt3 database.
cd /var/lib/postgres
pg_dump --clean --blobs --format=C -U rt3user rt3 >backup/rt3.‘date +%F‘
```

Import databáze

```
$ psql -f dumps/export.sql template1
$ pg_restore -v rt3.dump
$ pg_restore -C -d template1 rt3.dump
```

49.1.5.1.1. Převod a upgrade databáze

Na původním SQL serveru (`psql`) provedeme dump databáze, a tento si přeneseme na nový server.

```
/etc/init.d/apache stop
psql:~# su - postgres
postgres@psql:~$ pg_dump -Fp rt3 >rt3.‘date +%F‘.sql
```

* *# **pg_dump --clean --blobs --format=C -U rt3user rt3 >rt3.dump***

Na novém serveru (`rt`) vytvoříme uživatele rtuser

```
rt:~# su - postgres
postgres@rt:~$ psql template1
template1=# CREATE USER rtuser WITH PASSWORD ’********’ CREATEDB NOCREATEUSER;
template1=# \q
```

Na novém serveru:

```
postgres@rt:~$ createuser -AD rtuser
CREATE USER
postgres@rt:~$ createdb -E UTF-8 -O rtuser rtdb
CREATE DATABASE
postgres@rt:~$ sed -e ’s/rt3user/rtuser/g’ rt3.sql >rt.sql
postgres@rt:~$ psql rtdb -f rt.sql
```

49.1.6. Mail Gate

Dalším krokem bylo zajistit vstup mailů do RT. T.j. automatické zakládání tiketů zadaných mailem. Pro každý MTA se toto řeší jinak. Já jsem použil MTA Exim verze 3.36-13.

49.1.6.1. Exim 3 s použítím aliases

Nejjednodušší řešení je použít systémový `/etc/aliases` soubor. Do tohoto souboru přidáme řádky

```
### Request Tracker
rt:     "|/usr/bin/rt-mailgate --queue general --action correspond --url http://localhost/rt"
rt-comment:     "|/usr/bin/rt-mailgate --queue general --action comment --url http://localhost
```

Tyto řádky nám zajistí že maily adresované na `rt@rt.firma.cz` a `rt-comment@firma.cz` budou předány programu **rt-mailgate** spolu s dalšími parametry uvedenými na řádku. Program **rt-mailgate** pak přímým přístupem na web RT vytvoří z mailu tiket. Aby to fungovalo, je třeba ještě v konfiguraci eximu nastavit pod jakým uživatelem se bude program **rt-mailgate** spoouštět. V části `system_aliases` jsem tedy nastavil jako uživatele `nobody`. Po úpravě vypadá tato část v souboru `/etc/exim/exim.conf` následovně.

```
system_aliases:
  driver = aliasfile
  file_transport = address_file
  pipe_transport = address_pipe
  file = /etc/aliases
  search_type = lsearch
  user = nobody
```

49.1.6.2. Exim 3

* *section condition="author"*
* *rcsinfo="$Header: /home/radek/cvs/unix-book/input/unix/ch-request_systems.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $"*

Do konfiguračního souboru Eximu `/etc/exim/exim.conf` doplníme transportér

```
rt3_transport:
        driver = pipe
        user = nobody
        command = rt-mailgate \
                --action '${if def:local_part_suffix {comment}{correspond}}' \
                --queue "$local_part" \
                --url http://localhost/rt
```

A k tomuto transportéru napíšeme ovladač (director)

```
rt3_director:
        driver = smartuser
        suffix_optional
        suffix = -comment
        transport = rt3_transport
```

Ta se zdá že to nefunguje správně. V deníku se objeví následující hlášení:

```
2005-02-22 14:01:15 1D3Zf9-0000bt-00 <= radek@hnilica.cz H=sunrise.firma.cz [10.16.66.18]
 P=esmtp S=755 id=E1D3ZW7-0008G1-00@yoda.hnilica.cz
2005-02-22 14:01:18 1D3Zf9-0000bt-00 == rt@rt3.firma.cz T=rt3_transport defer (0): Child
 process of rt3_transport transport returned 75 (could mean temporary error) from command:
 /usr/bin/rt-mailgate
2005-02-22 14:01:18 1D3Zf9-0000bt-00 failed to open DB file /var/spool/exim/db/retry: File exists
```

49.1.6.3. Apache

Zajisté vás napadne, co dělá konfigurace Apache v sprovozňování RT-MailGate. Je to proto, že **rt-mailgate** předává požadavky došlé poštou do RT přes jeho webové rozhraní. Konkrétně přistupuje k url `REST/1.0/NoAuth/mailgate`. Chvíli mi trvalo, než jsem to celé pochopil. Protože používám pro kontrolu přístupu do RT LDAP, a celé je to ošetřuji v Apache, narazil jsem na problém. Apache vyžadoval ověření uživatele i pro **rt-mailgate**. Nakonec pomohl přidat za sekci `<Location /rt>` další sekci, která realizuje výjimku v ověřování uživatele.

```
<Location /rt/REST/1.0/NoAuth>
        AuthName        None
        Satisfy         Any
        Order           Deny,Allow
        Allow From      localhost
        Deny From       All
</Location>
```

49.1.7. Čeština

Tak jak RT nainstalujeme, můžeme používat češtinu. Ve skutečnosti pracuje RT interně v UTF-8 a jeho www rozhraní bylo napsáno pro mnoho jazyků. RT sám detekuje preferovaný jazyk od našeho prohlížeče a stránku nam dodá v tomto jazyce. Problém češtiny je v tom, že odesílané emaily jsou kódovány v UTF-8. Náš oblíbený

vnitrofiremní poštovní klient Pegasus Mail však s UTF-8 moc nespolupracuje. Nová verze která má být uvolněna každou chvíli (je 2005-02-23) již UTF-8 má umět. Prozkoumáním konfiguračního souboru jsem přišel na dvě nastavení které ovlivňují vstupní a výstupní mailové kódování. Tyto jsem přepl do preferovaného iso-8859-2.

```
# Cestina v iso-8859-2
@EmailInputEncodings = qw(utf-8 iso-8859-2 us-ascii);
Set($EmailOutputEncoding , 'iso-8859-2');
```

49.1.8. Použití

* *section id="rt.usage"*
* *rcsinfo="$Header: /home/radek/cvs/unix-book/input/unix/ch-request_systems.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $"*

Abych osvětlil k čem program Request Tracker je, popíši jeho užití v několika jednoduchých scénářích.

49.1.8.1. Závada tiskárny

Nejdříve jednoduchý modelový postup řešení závady/problému na tiskárne `laser5`.

1. Na tiskárně `laser5` nastala závada. Uživatel, nemoha tisknout, zavolá na oddělení IT a vysvětlí svůj problém. Pracovník IT který hovor přijímá, vyslechne uživatele a vytvoří nový lístek popisující závadu.
2. Technik VT, jenž se zabývá drobnými problémy a závadami techniky je RT systémem informován o novém lístku. Protože má právě čas, přečte si popis uživatelova hlášení a v systému RT nastaví sebe jako vlastníka lístku.
3. Poté se odebere k tiskárně aby zjistil technické podrobnosti závady a případně ji na místě odstranil.

   Tady může být ukončen životní cyklus lístku. V případě že se jednalo o drobnou závadu jenž je možno na místě odstranit, technik tuto odstraní, a v systému RT napíše řešení (resolution). Lístek v RT je označen jako vyřešený, zmizí z fronty lístků a všichni zainteresovaní jsou dopisem informováni o odstranění závady.
4. V případě že závada je vážná a vyžaduje servisní zásah, život lístku pokračuje dál. Po ohledání závady technik napíše komentář k závadě, a informuje servis o potřebě zásahu. Lístek jako živý (nevyřešený) dále zůstává ve frontě.
5. Při příjezdu servisního technika je možno vytisknout historii lístku se všemi odpoveď mi a komentáři. Po odstranění závady servisním technikem pracovník IT o tomto sepíše zápis (řešení) a lístek je uzavřen.

49.1.8.2. Bananamaster 3000

Tato ukázka je z dokumentace RT uveřejněné na About RT (http://wiki.bestpractical.com/index.cgi?ManualIntroduction).

1. Billy má problém se svým Bananamaster 3000, takže pošle zprávu na <`help@example.com`>.
2. RT zazanemná novou událost do své databáze a vrátí Billimu elektronickou potvrzenku jenž obsahuje číslo vytvořeného tiketu a v ideálním případě nějaké pomocné informace. Billy se může (měl by) v budoucí komunikaci odkazovat na číslo tiketu které obdržel.
3. RT přepošle Billiho zprávu k řešení personálu.
4. Jeden ze zaměstnanců, Jane, je firemní odborník na Bananamaster 3000. Takže Jane převezme Billiho žádost a napíše odpověď. RT ji automaticky zaznamená a pošle Billimu.

5. Billy je spokojeným zákazníkem; Jane vyřešila jeho problém a může jít na pivo.

Toto workflow je flexibilní, může být upraveno ve vaší organizaci jiným způsobem, a nepotřebujete do něj zahrnovat banány.

49.1.9. Administrace

* *rcsinfo="$Header: /home/radek/cvs/unix-book/input/unix/ch-request_systems.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $"*

Zde se primárně oritentuji na adminstrční zásahny a opravy přímo v SQL databázi. U některých prblémů jsem nepřešel na to jak je řešit z Web rozhraní.

49.1.9.1. Ztráta hesla / obnovení přístupu

Pokud se nám stane, že zapomeneme heslo do Request Trackeru a to heslo uživatele root, můžeme si nastavit heslo přímým zásahem do databáze. Musíme tedy mít heslo k databzi, které je ovšem poznačen v konfiguračním souboru.

V případě PostgreSQL použijeme postup:

```
# su - postgres
$ psql rtdb
Welcome to psql 7.4.7, the PostgreSQL interactive terminal.
...
rtdb=# update only users set password=md5('heslo') WHERE name='root';
UPDATE 1
rtdb=# \q
```

49.1.9.2. Účty a uživatelé

* *rcsinfo="$Header: /home/radek/cvs/unix-book/input/unix/ch-request_systems.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $"*

FIXME:

Chceme-li uživateli povolit změnu jeho kontaktních údajů a hesla, musíme mu dát právo `ModifySelf` Nastavení provedeme v Configuration⟶Global⟶User Rights

49.1.9.3. Přístupová práva

* *rcsinfo="$Header: /home/radek/cvs/unix-book/input/unix/ch-request_systems.xml,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $"*

Systém přístupových práv je poměrně rozsáhlý, proto se jej pokusím zmapovat. Přístupová práva nastavujeme na řadě míst a je lehké ztratit přehled. Nejdříve popíši jednotlivé práva která můžeme přidělit.

AdminQueue

FIXME:

49.1.10. Poznámky

V RT3 je definována proměnná `$RT::WebExternalAuth`

Ke dni 2005-02-22 jsou v Debian GNU/Linux balíčky:

- `request-tracker1` — (Woody) verze 1.0.7-2
- `request-tracker3` — (Sarge, Sid) verze 3.0.12-6
- `request-tracker3.2` — (Sid) verze 3.2.2-3
- `rt3-clients` — (Sarge) verze 3.0.12-6

49.1.11. Postupy

49.1.11.1. Založení nové fronty

Projdeme menu: Configuration⟶Queues⟶New queue a na formuláři vyplníme pole následovně:

- Queue Name: — jméno nově zakládané fronty
- Description: — uvedeme krátký popis k čemu je fronta určena
- Correspondence Address: — vstupní mail adresa pro zakládání tiketů
- Comment Address: — vstupní mail adresa pro komentáře

Dale pokračujeme a v konfiguraci nově založené fronty nastavíme Watchers, Group Rights.

49.2. phpGroupWare

Dalším horkým kandidátem je phpGroupWare (http://www.phpgroupware.org). Tento softwér je však úplně jiného druhu než Request Tracker. Je podstatně rozáshlejší a počítá s jiným nasazením.

49.3. Double Choco Latte

FIXME: prozkoumat program a dopsat sekci

VII. X Window

X Window a vše co s tím souvisí. Tedy programy, nástroje, použití. Jen programování se nachází v kapitole 68

Kapitola 50. Uvedení do problematiky, aneb jak to všechno funguje

abstract

Blockquote. Vysvětlení základních principů na kterých je system X Window postaven.

TBD

* ***WORKING: Editovat.***

Obyčejný text.

Kapitola 51. Xorg

- Nastavení Xorg (http://debian.nfo.sk/Wiki/NastavenieXorg)

Kapitola 52. Instalace nvidia kernel modulu

```
# aptitude install linux-source-2.6.24
```

Kapitola 53. Ovladače grafických karet

53.1. Instalace kernelového modulu pro karty NVidia

Odkazy:

- Diskuse (http://www.root.cz/diskuse/1307/) k článku Debian Etch a nvidia drivery na Root.CZ
- Simple instructions on how to install nVidia drivers for X11 on Debian written by Len Sorensen (http://desiato.tinyplanet.ca/~lsorense/debian/debian-nvidia-dri-howto.txt)
- NvidiaGraphicsDrivers (http://wiki.debian.org/NvidiaGraphicsDrivers) on Debian Wiki
- nvidia debian testing kernel 2.6.24 possibilities (http://www.linuxquestions.org/questions/debian-26/nvidia-debian-testing-kernel-2.6.24-possibilities-623754/)

Předpokládejme že mám nainstalované jádro `linux-image-2.6-686-bigmem`. K tomuto jádru doinstaluji hlavičkové soubor.

```
# aptitude install linux-headers-2.6-686-bigmem
```

Jestli nemáme nainstlujeme module-assistant.

```
# aptitude install module-assistant
```

Nyní si necháme vytvořit balíček s ovladačem.

```
# m-a a-i nvidia
```

To be done:

Kapitola 54. Základní nástroje a programy

Program **xwd** umožňuje dělat dumpy oken.

Kapitola 55. Programy pro grafická prostředí

- update-notifier (http://packages.debian.org/etch/update-notifier) (Debian Etch)
- adept-notifier (http://packages.debian.org/etch/adept-notifier) (kde, Debian Etch)

Kapitola 56. Xfce

Prostředí Xfce a jak si je vyladit.

Pokud instalujeme systém, můžem instalovat z CD označeného `xfce`.

Pokud již systém máme nainstalován, můžeme doinstalovat jen xfce.

```
# aptitude install xfce4
# aptitude install xfce4-goodies
```

56.1. Instalace a aktualizace programů

Protože holá instalace Leny β1 xfce nemá vůbec v měnu žádné možnosti okolo upgrade software, musíme si sami jednotlivé balíčky doinstalovat. Prvním z potřebných balíčků je program pro správu balíčků a jejich instalaci. Rozhodl jsem se použít synaptic takže jsem jej z konzoly nainstlaoval

```
# aptitude install synaptic
```

Další instalace již mohu dělat pomocí Synapticu ale zde je uvádím v zápise pro **aptitude**.

Další jsou automatické instalace aktualizací. Ty provádí applet `apt-watch-gnome`. Alespoň jsem zatím nenašel jiný. Abychom ho mohli použít, nainstalujeme k němu `xfce-xfapplet-plugin`.

```
# aptitude install apt-watch-gnome xfce4-xfapplet-plugin
```

Vybereme si lištu na které aplet chceme mít, a dáme přidat novou položku. Vybereme XfApplet a ten se nás zeptá, který že to gnome applet chcemem. Zde vybereme apt-watch-gnome. Nezbývá než nakonfigurovat podle vlastní potřeby, například nastavit jak často se aktualizace kontrolují, a které typy aktualizací se vlastně sledují.

> **Poznámka:** Existuje balíček pro gnome, `update-manager`, původem z Ubuntu. Tento můžeme s výhodou také použít.
>
> ```
> # aptitude install update-manager
> ```

56.2. Vypínání počítače

Aby fungovalo vypínání počítače přímo z Xfce, musím být oprávněn jako uživatel spouštět program **xfsm-shutdown-helper**. Toho dosáhnu přidáním následujícího řádku do /etc/sudoers. Připomínám že tento soubor se edituje příkazem **visudo**.

`uživatel stroj`=/usr/sbin/xfsm-shutdown-helper

V mém konkrétním případě, kdy můj login je `radek` a počítač se jmenuje `fram` to bylo:

```
radek fram=/usr/sbin/xfsm-shutdown-helper
```

56.3. Menu přidat/ubrat programy

Tuto položku do menu lze přidat až od Lenny.

```
# aptitude install gnome-app-install
```

56.4. Další balíčky

- notification-daemon-xfce (http://packages.debian.org/search?keywords=notification-daemon-xfce) (Debian Lenny)
- xfce4-appfinder ()
- xfce4-xfapplet-plugin ()
-
-

Kapitola 57. Gnome

57.1. Zajímavé applety

Pro sledování aktualizací a "automatickou" aktualizaci máme k dispozici applet `apt-watch-gnome`. Tento umístníme na lištu a nakonfigurujeme podle potřeby. Applet sleduje jsou-li nové aktualizace programů jenž máme nainstalovány. Instalaci aktualizací provádí přes nakonfigurovany manažer balíčků. K dispozici jsou možnosti: `synaptic`, `aptitude`, `apt-get`, `gnome-apt`, nebo vlastní.

VIII. Programování pod UNIXem, a Linuxem obzvláště.

Všechno kolem programování a skriptování.

Kapitola 58. Jak psát dobrý kód

Kapitola věnovaná způsobům programování.

58.1. Defensivní programování

Defensivní programování je styl, kdy nic nepředpokládáme. Tedy nemáme žádné dobré předpoklady jako že uživatelé budou zadávat správné vstupní hodnoty, že technická zařízení se vždy budou chovat jak bylo popsáno v návodu, že se skrátka všechny věci budou dít tak aby náš program fungoval.

Neexistenci pozitivních předpokladů nahradíme jedním jediným negativním předpkladem.

> Vše co se může pokazit, se pokazí.

Kapitola 59. Příkazová řádka

Command Line Interface

Základní komunikace Linuxu a UNIXu obecně mezi uživatelem a počítačem se odehrává na terminálu přes tak zvanou příkazovou řádku. Jedná se o některý z existujících programů jenž nazýváme shell který na terminálu očekává příkazy a vypisuje jejich výsledky.

Nejčastěji používaným shellem je Bash.

Nastavení proměnné PS1.

Kapitola 60. bash

* *chapter id="bash" xreflabel="Bash"*

Odkazy a zdroje:

- Advanced Bash-Scripting Guide (http://www.tldp.org/LDP/abs/html/index.html)
- Bashish (http://bashish.mine.nu/BashishWiki/index.php/Main_Page)
- BASH Programming - Introduction HOW-TO (http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html) by Mike G
- Bash Guide for Beginners (http://tldp.org/LDP/Bash-Beginners-Guide/html/) by Machtelt Garrels
- Bash Hackers Wiki (http://bash-hackers.org/wiki/doku.php/start)
- Grouping commands (http://bash-hackers.org/wiki/doku.php/syntax/ccmd/grouping_plain)
- bashscripts.info (http://www.bashscripts.info)

Bash je program ze skupiny programů nazývaných shelly. K této skupině patří programy jako **ash**, **sh**, **ksh**, **pdksh**, **zsh**, **tcsh** a mnohé další. Účelem těchto programů je vytvořit rozhraní mezí člověkem, operátorem na jedné straně a operačním systémem počítače na straně druhé. Je to jakási skořápka (shell) která obaluje operační systém a dovoluje interakci s člověkem.

Ve své základní funkci dovolují shelly spouštět ostatní programy. Dovolují seskupovat programy do celků, spoštět jeden na základě výsledků předchozího či opakovaná vícenásobné spuštění stejného programu. Lze vytvářet dávky příkazů od jednodušších až po složitější jenž mají chování samostatných programů.

FIXME: TBD:dopsat.

Programování v shellu, je velmi specifické. Shell jako takový je určený k běžnému a soustavnému používání, spouštění programů a vytváření vezeb mezi těmito programy. Slouží tedy hlavně jako lepidlo, jako tmel, který spojuje jednotlivé programy do funkčního celku o vyšší kvalitě.

Specifičnost shellu se odráží právě ve schopnosti spouštět a spojovat programy. Za tímto účelem má shell řadu nástrojů se kterými se seznámíme.

Tyto programy slouží jako interprety příkazů. Tvoří rozhraní mezi operačním systémem a člověkem jenž sedí u terminálu. Přihlásíme-li se do počítače a pracujeme, pracujeme vlastně se shellem. Na jedné straně nám shell dovoluje zadávat příkazy, skupiny příkazů, a na druhé straně je to silný programovací jazyk. **FIXME:**

shell — skořápka

Shell je lepidlo/tmel jímž spojujeme do funkčních celků stávající programy a vytváříme tak programy nové se specifickou funkcí.

Proč programovat v shellu? Inu nemusíme přímo programovat, ale shell je mocný nástroj jímž zvládneme vyskriptovat řadu jednoduchých problémů aniž bychom museli sáhnout po jiném programovacím jazyku jako je například C.

60.1. Podmíněné příkazy

Jedná se o příkazy s podmínkou. Po vyhodnocení podmínky se vykoná další část příkazu.

```
if condition; then command; fi
if condition; then command1; else command2fi
if condition1; then
    command1
elif condition2; then
    command2
else command-else
fi
```

```
case value in
template1) command1;;
template2) command2;;
*)         command-else;;
esac
```

60.2. Funkce

```
function mojeFunkce() {
    # tělo funkce
}
```

Když potřebujeme zjistit, je-li nějaká funkce definována, můžeme využít příkazu **declare**.

```
if declare -F|grep "setUp$"; then
    eval setUp
fi
```

Můžeme zkusit použít i příkaz **type**.

```
if type $fce >/dev/null 2>&1; then
    $fce
fi
```

60.3. Parametry skriptu a jejich analýza

Odkazy:

- http://www.ibm.com/developerworks/linux/library/l-bash-parameters.html

Spuštěný skript má k dispozici řadu proměnných ve kterých jsou informace o parametrech a přepínačích uvedených na příkazové řádce. Jsou to zejména:

`$0`

Název a cesta k spuštěnému skriptu.

`$1`, `$2`, `$3`, ...

Proměnné obsahující jednotlivé parametry tak jak jsou uvedeny na příkazovém řádku. Takto jednoduše se lze odkazovat na parametry od 1. po 9. Na další parametry se odkazujeme zápisem `${10}`, `${10}`, ...

`$*`

Všechny parametry počínaje prvním, tedy `$1`, `$2`, `$3`, ...

`$@`

Všechna parametry počínaje prvním. Pokud je tato proměnná použita v uvozovkách `"$@"`, tak se expanduje na `"$1" "$2" ...`

`$#`

Celkový počet parametrů, bez parametru `$0`

Uvedené proměnné mají uvedené hodnoty definovány jen ve skriptu mimo těla funkcí. V tělech funkcí nabývají hodnot podle parametrů předávaných funkci.

Argumenty příkazové řádky můžeme procháze jednoduchým způsobem cyklem **for** s vnořeným příkazem **case**.

```
for a in "$@"; do
        echo "process argument $a"
        case "$a" in
        -a)
                VOLBA_A=1
                ;;

                ...
        esac
done
```

Pro sofistikovanější analýzu je vhodné použít k tomu existující nástroje. Tyto jsou popsány dále. Jedná se zejména o vnitřní příkaz bashe **getopts** a externí program **getopt**.

60.3.1. Použití interního příkazu bashe getopts

Odkazy:

- SysAdmin to SysAdmin: More power with bash getopts (http://www.linux.com/articles/113836) by Brian Jones on linux.com

Pro analýzu parametrů můžeme s výhodou použít příkaz **getopts**. Uvedu vzorové použití:

```
while getopts "an:p:hv" optname; do
    case "$optname" in
        "v")
     ssversion="$OPTARG"
     ;;
        "a")
     echo "Option $optname is specified"
     ;;
 "n")
     echo "Option $optname has value $OPTARG"
     ;;
        "p")
     echo "Option $optname has value $OPTARG"
     ;;
        "h")
     echo "Option $optname is specified"
     ;;
         *)
     errormsg="Unknown parameter or option error with option - $OPTARG"
            echo "Unknown error while processing options"
            ;;
    esac
```

```
    echo "OPTIND is now $OPTIND"
done

while getopts "h" OPTIONS; do
    case ${OPTIONS} in
        h|-help) echo "${usage}";;
    esac
done

while getopts ":u:a:s:v" options; do
    case $options in
    u) uname=$OPTARG;;
    a) attrs=$OPTARG;;
    s) searchattr=$OPTARG;;
    v) att=ALL;;
    h) echo $usage;;
    \?) echo $usage
        exit 1;;
    *) echo $usage
       exit 1;;
    esac
done
```

60.3.2. Externí program getopt

Odkazy:

- Getopt and getopts (http://aplawrence.com/Unix/getopts.html) by A.P. Lawrence

Program je běžnou součástí Linuxových distribucí, například v Debianu je v balíčku util-linux. Jeho předností, oproti jiným způsobům analýzy příkazové řádky je snadná specifikace nejn krátkých přepínačů jako `-a` ale i dlouhých přepínčů jako například `--help`. Proto jej mám velmi v oblibě a používám velmi často.

Po nějaké době používání, a vylepšování jsem přišel s následujícím použitím.

```
#!/bin/bash
# -*- coding:utf-8; -*-
# $Id: ......$
# Jednořádkový popis programu/skriptu.
# Copyright (c) 2009 Já První Největší
# Specifikace práv jako: Všechna práva vyhrazena.  Nebo dělejte si každý co chcete.

declare -r COPYRIGHT="(c) 2009 Já První Největší
declare -r PROGRAM_NAME=${0##*/}
declare -r PROGRAM_VERSION="2.3"
declare -r PROGRAM_REVISION=$(echo '$Revision: 1.52 $' | awk '{print $2}')

declare -r DESCRIPTION="

Víceřádkový popis programu a jeho použití.  Toto je taková malá dokumentace.
Neplet'te si prosím s popisem argumentů příkazové řádky.  Tento a podobné
informace jsou poskytovány funkcí echo_usage.

"
```

```
# Základní kontrola jestli spouštíme program na správném počítači/serveru.
# to ve většině případů opravdu nepotřebujete.  Ale někdy se to velmi hodí.

# Nahrání knihoven a základní nastavení prostředí.  Například:
source /usr/local/sad/lib/environment
source /usr/local/sad/bin/load-vars
declare -ir TRUE=0 FALSE=1      # Boolean variables
PATH=/usr/local/sad/bin:$PATH

echo_version() {
    echo "$PROGRAM_NAME $PROGRAM_VERSION rev $PROGRAM_REVISION  $COPYRIGHT"
}

echo_usage() {
    echo_version
    echo "usage: $PROGRAM_NAME [-dvh] ..."
    echo -e " -d,--debug\tdisplay debug messages, increase debuging level"
    echo -e " -v,--verbose\tbe more verbose, increase verbosity level"
    echo -e " -h,--help\tdisplay this little usage help"
    ...
}


echo_description() {
    echo "$DESCRIPTION"|grep -v "^ *$"
}

# Argument parser

declare rest_args=
declare -i option_verbose=$FALSE verbosity_level=0
declare -i option_debug=$FALSE debug_level=0
declare -i option_....

function parse_arguments() {
    local option
    local -i fatal=$FALSE
    eval set -- "$arguments"

    while true; do
        option=$1; shift
        case "$option" in
            -d|--debug)         option_debug=$TRUE; ((debug_level++));;
            -v|--verbose)       option_verbose=$TRUE; ((verbosity_level++));;
            ...
            -V|--version)       echo_version; exit 0;;
            -D|--description)   echo_description; exit 0;;
            -h|--help)          echo_usage; exit 0;;
            --) rest_args="$@"; break;; # Stop parsing on '--' mark.
            *)  echo "$PROGRAM_NAME: Argument parsing error: ($option): $*"
                exit 1
                ;;
        esac

        # Check if mandatory parameters were given
        ...
```

```
            if ((fatal)); then
                echo "$PROGRAM_NAME: There was an fatal error.  Exitting!"
                exit 1
            fi
        done
    } # parse_arguments()

    ### Startup code. ###############################################
    arguments=$(getopt -u -n $PROGRAM_NAME \
        -o "dhqsvVD" \
        -l "debug help silent verbose version description" \
        -- "$@")
    parse_arguments

    if ((option_debug)); then
        echo_version
        set -x
    fi

    ########   M A I N   ########################################
```

Teď se podíváme jak to celé funguje. Na začátku programu jsou úvodní komentáře. Navykl jsem si na tento několikařádkový blok a dávám jej všude standardně. Ve zkratce nás informuje o tom že tento program je v bashi. To je první a jediný povinný řádek. Poté následuje řáde s informacemi o verzi a souboru ze systému správy zdorjového kódu jako je RCS/CVS/SVN/... Navykl jsem si vždy nějaký systém použít. Alespoň RCS pokud nepouiji jiný.

Pořád se učím nové věci, ještě nedávno jsem předával argumenty do funkce `parse_arguments` jiným způsobem. A ten fungoval dlouho. Pak se ale vyskytl speciální případ a já zjistil že to tak nejde. Na třetím řádku je jednořádkový popis programu. Je to velmi dobré pro rychlou orientaci. Název programu, ačkoliv by měl být dostatečně popisný, takový v řadě případů nemusí být. Poté následují řádky o Copyrightu a za nimi krátká informace o licenci.

Další částí v programu je blok konstant obsahující informace o samotném programu. Tento blok rovněž používám ve všech programech psaných v bashi. Tyto konstanty, velmi popisné, jsou pak v následujících funkcích použity. Definuji zde konstantu `COPYRIGHT` s informacemi o držiteli kopyrajtu. Dále `PROGRAM_NAME` jenž obsahuje jméno programu. To získám z parametru *`$0`*. Informaci o verzi programu `PROGRAM_VERSION`, udržovanou ručně. Poslední je informace o revizi `PROGRAM_REVISION` získanou z systému správy verzí pokud to umí. Určitě to umí RCS a CVS.

Poté definuji konstantu `DESCRIPTION` která obsahuje víceřádkový popis programu. Zastávám názor že dokumentace programu umístněná přímo do programu je velmi dobrá věc. Vždy ji máte po ruce a nemusíte ji hledat. A vždy ji píšu. Zde je účel programu to k čemu je určen a souvislosti s ostatními programy a vztahy k dalším částem projektu jehož je program součástí.

Pak dávám do programu základní kontrolu. Některé programy, jako součásti větších projektů jsou určeny jen k běhu na určených serverech. Spouštění na jiných strojích nemá smysl, nebo může být i nebezpečené. Zejména pokud program běží s právy roota. Například v projektu určeném k běhu na desítkách až stovkách počítačů kdy většina je „satelitních“ a pak jsou servery „distribuční“ a servery „centrální“ některé programy smí být spouštěny jen na určitém typu serveru. Tato kontrola podle různých znaků, jako je přítomnost konfigurace satelitního serveru, nebo přítomnost konfigurace distribučního serveru, pokud zjistí že je spuštěna na nesprávném serveru vypíše krátkou informaci a ukončí program.

Poté následuje blok příkazů pro vytvoření základního prostředí. Zde nahrávám příkazem **source** knihovny funkcí a proměnné specifické pro projekt. Program například vůbec neví, kde se nachází adresář s proměnnými. Pokud nějakou chce použít prostě použíje konstantu `$VARIABLES` již správně defunije nějaká knihova ze zde

právě nahraných. V této části rovněž defunuji logické konstanty TRUE a FALSE. A podle potřeby opravím či pozměním proměnnou `PATH`.

Dále následují tři funkce. `echo_version`, `echo_usage` a `echo_description`. První a poslední jsou všude stejné a využívají informací definovaných na začátku programu v konstantách popisujících program. Funkce `echo_usage` zobrazuje informace o paramerech v příkazové řádce. Obsah této funkce je specifický pro každý program.

Nyní následuje definice proměnných po jednotlivé volby v příkazovém řádku a samotná funkce `parse_arguments` která je naplní podle toho co v příkazovém řádku zadáme.

* *Podrobněji popsat parse_arguments()*

Náslduje startovací kód programu. Zde zavoláme externí program pro analýzu příkazové řádky **getopt** s parametry popisujícími možné přepínače a informace o těchto přepínačích. To jsou písmenka za přepínačem `-o` a slova dlouhých parametru za přepínačem `-l`. Poté zavolám samotnou funkci `parse_arguments` která podle informací v proměnné `arguments`, získané voláním **getopt**, nastaví proměnné `option_...` a další. V programu se dále používají jen tyto proměnné.

Posledním standardním kódem který používam je přepnutí do ladicího režimu v závislosti na hodnotě proměnné `option_debug` získané z přepínače `-d` nebo `--debug`. Zde končí šablona a dále následuje samotný program.

Pokud se vám tato šablona zdá dlouhá, nemusíte ji používat. Je v ní jen má osobní zkušenost. Budete li programovat projekt s větším počtem skriptů, jenž jsou navzájem závislé a propojené, tato nebo nějaká jiná šablona vám bude velmi k užitku.

* ***FIXME:****Odstranit zbytek sekce.*

```
function parse_arguments() {
    while true; do
        option=$1; shift

 case "$option" in
            -h|--help|-?)       usage; exit 0;;
            -i|--interactive)   INTERACTIVE=1;;
            -c|--count)         COUNT=$1; shift;;
            ...
            --)                 break;;
            *) echo "Argument parsing error: ($option): $*"; exit 1;;
        esac
    done

    # Argument checking
    ...
}
⋮
parse_arguments $(getopt -n "${0##*/}" -o "abc:hi" -l "help interactive count:" -- "$@")
```

60.3.2.1. Poznámky

```
#!/bin/sh
# File: gg
# Usage:  gg -avp arg1 arg2
max=0
while getopts "eprvh" flag
do
        case $flag in
        ... (omitted)
```

```
        esac

        if [ $OPTIND -gt $max ]
        then
                #echo "setting max to $OPTIND"
                max=$OPTIND
        fi
done
shift $max

# now $@ just has what's left.
```

60.3.3. Externí skript getoptx

Odkazy:

- Bash long options shell function **getoptx** (http://nlp.cs.jhu.edu/~edrabek/utils/)

FIXME:

60.3.4. Nápověda k parametrům programu

Nejlépe si připravit pro zobrazení nápovědy funkci kterou pak v okamžiku analyzování příkazové řádky můžeme zavolat. Obvyklým názvem pro tento malý help je `usage`. Jako funkci si ji pak pojmenuju například `display_usage` nebo `echo_usage`.

```
usage() {
        cat <<EOF
usage: $0 [-htrpv]

This script run the test1 or test2 over a machine.

OPTIONS:
  -h\tShow this message
  -t    Test type, can be test1 or test2
  -r    Server address
  -v    Verbose
EOF
}
```

60.4. Čtení vstupu

Pro čtení ze vstupu má bash příkaz **read**. Tento nám dovoluje přečíst údaje ze vstupního proudu (stdin) a načíst je do proměnných bashe. Jeho základní použití je:

```
$ read prvni druhy
hello world
$ echo $prvni
hello
$ echo $druhy
```

```
world
$
```

Jak je na uvedeném příkladu vidět, použití **read** je jednoduché, příkaz přečte „větu“ ze standardního vstupu (stdin) a rozdělí ji podle oddělovače do jednotlivých proměnných.

Oddělovač je znak, nebo množina znaků, které oddělují jednotlivá pole vstupní věty. Jako standardní oddělovač používá **read** znaky mezera, tabulátor a znak konce řádku. Pokud chceme použít jiný, sdělíme to příkazu **read** nastavením proměnné `IFS`.

```
$ IFS=" :=\t" read name value rest
NAME: radek
$ echo "$name='$value'", $rest
NAME='radek'
```

Proměnná `IFS` má na rozdělování věty na slova (parsing) složitější vliv. Algoritmus rozdělování se chová jinak k regulérním znakům, jako jsou v našem případě ":" a "=" a jinak k bílým znakům jako je mezera `" "`, tabulátor `"\t"` a znak konce řádku `"\n"`. Vyzkoušejte si různé kombinace znaků v IFS a experimentujte i se vstupy. Další informace k rozdělování vstupní věty najdete v části "Word Splitting" manuálové stránky programu bash.

60.4.1. Příklady

Různé příklady.

Následující dva příklady jsem objevil na http://www.unixguide.net/unix/bash/E4.shtml.

```
read A B C D << HERE
    $(IFS=.; echo $(/usr/local/bin/ipaddr))
HERE

read A B C D < <(IFS=.; echo $(/usr/local/bin/ipaddr))
```

60.5. Čtení souborů

Pokud chceme zpracovávat celý soubor, řádku po řádce, můžeme použít následující jednoduchý cyklus s přesměrováním standardního vstupu.

```
#!/bin/sh
while IFS=" :=\t" read name value
do
    echo "$name='$value'"
    ...
done < /cesta/k/souboru
```

Uvedený příklad nám poslouží jen v jednodušších případech. Ve složitějších narazíme na omezení přesměrování standardního vstupu. Pokud potřebujeme načítat soubor v několika příkazech **read** a ukládat si zjištěné informace do proměnných, s uvedeným mechanismem nevystačíme. Problém na který narazíme je problémem vnořených shellů. Aby všechny příkazy **read** četly z jednoho souboru, musíme přesměrování vstupu učinit uzavřením celého bloku příkazů do vnořeného shellu pomocí kulatých závorek. Vstupní soubor pak na konci přesměrujeme do tohoto bloku. Uvnitř vnořeného shellu ale není možno modifikovat proměnné vně. Takže pokud budeme provádět například vyhledávání maxima či výpočet součtu hodnot. S následující konstrukcí neuspějeme.

```
...
maximum=0
```

```
(
    ...
    read line
    ...
    while ... read name value ...
        ...
        maximum=...
    done
) < $IFILE
# maximum je zde 0
```

Řešením této situace je využití možnosti příkazu **read** načítat vstupní větu z jiného deskriptoru než standardního vstupu. Takto můžeme konstruovat mnohem složitější algoritmy kdy čteme z více souborů najednou a čtená data kombinujeme. Nejdříve tedy jak bude vypadat náš jednoduchý příklad.

```
exec 3</cesta/k/souboru
while IFS=" :=\t" read -u3 name value; do
    echo "$name='$value'"
    ...
done
exec 3<&-
```

První řádek otevírá soubor v deskriptoru 3. Volba `-u3` říká **read** že má číst z deskriptoru 3 a nikoliv ze standardního vstupu (deskriptor 0). Poslední příkaz uzavírá soubor v deskriptoru 3.

Jště uvedu ukázkový příklad čtení konfiguračního souboru.

```
CONF=/cesta/k/souboru
exec 3<$CONF
while read -u3 line; do
    if [[ "$line" =~ "^#" ]]; then continue; fi  #Ignorujeme komentáře
    if [[ "$line" =~ "^[ \t]*$" ]]; then continue; fi  #Ignorujeme prázdné řádky
    ...
done
exec 3<&-
```

Dříve jsem nevěděl o možnosti zadávat v příkazu **read** číslo deskriptoru, a tak jsem používal následující vzor. Na začátku si uložím stdin, otevřu v stdin soubor a na konci zase obnovím stdin. Toto dělají dva příkazy **exec**.

```
#!/bin/sh
IFILE=/etc/passwd
exec 3<&0 <$IFILE
while IFS=":" read login hash uid gid gecos home shell; do
    echo "$login $shell"
done
exec 0<&3 3<&-
```

60.6. Parsování textu

Bash nemá mnoho nástrojů pro parsování textu, ale příkaz read se dá s úspěchem použít. V kombinaci s přesměrováním vstupu <<< snadno rozdělíme jeden řádek textu podle požadovaného oddělovače slov.

```
IFS=":" read prvni druhy zbytek <<<"text"
```

60.7. Práce s SQL databázemi

S databázemi se pracuje s pomocí klientů těchto databází. Bash nemá žádné nástroje jak s nimi pracovat přímo, ani žádné knihovny. Jediným způsobem jak pracovat s databázemi je volat klienta těchto databází.

```
odpoved=$(mysql -e "SQL příkaz")
```

nebo

```
odpoved=$(mysql <<EOF
SQL příkazy
EOF
)
```

Poznámka: Teoreticky je možno spusti databázového klienta na pozadí a číst jeho výstup.

60.8. Datum a čas

Odkazy:

- O příkazu date na Wikipedii (http://en.wikipedia.org/wiki/Date_(Unix))
-

Bash nemá žádné zabudované nástroje pro práci s datumy a časem. Pro tyto účely využíváme externí programy jako **date**, **touch** a jiné. A dále využíváme možností práce s textem které bash nabízí.

60.8.1. Porovnání datumů/časů

Tedy zjištění v jakém vzájmeném vztahu jsou dva datumy. Tedy který je nižší (starší) a který vyšší (mladší). Pro tento úkol se nabízejí dvě cesty řešení.

Program **test** a jeho zabudovaný eqvivalent v bashi má možnost porovnávat dva soubory podle data poslední modifikace. Jedná se parametry `-nt` a `-ot` zabudovaného porgramu **test**.

```
if [ file1 -nt file2 ]; then ... fi
```

Pokud časy které potřebujeme porovnávat jsou časy modifikace souborů, můžeme přímo použít program test.

```
if [ file1 -nt file2 ]; then
    echo "Soubor file1 je mladší a soubor file2 je starší."
else
    echo "Soubor file1 je starší a soubor file2 je mladší."
fi
```

Možnosti které nám tato srovná umožňují ukazuje názorně následující program. Experimentujte s prvním parametrem a sledujte jak se chová.

```
#!/bin/sh
case $1 in
    1) touch file1 file2;;
    2) touch file1; sleep 1s; touch file2;;
    3) touch file2; sleep 1s; touch file1;;
```

```
    *) echo "zadej parametr 1, 2 nebo 3"; exit 1;;
esac
ls -l file[12]

if [ file1 -nt file2 ]; then
    echo "čas_modifikace(file1) > čas_modifikace(file2)"
else
    echo "čas_modifikace(file1) <= čas_modifikace(file2)"
fi


if [ file1 -ot file2 ]; then
    echo "čas_modifikace(file1) < čas_modifikace(file2)"
else
    echo "čas_modifikace(file1) >= čas_modifikace(file2)"
fi

if ! [ file1 -nt file2 ] && ! [ file1 -ot file2 ]; then
    echo "Oba soubory jsou stejně staré."
fi
rm file[12]
```

Pokud máme čas/datum v řetězci a nejedná se o datum modifikace souboru, pomůžeme si malým trikem za použití programu **touch**. Tento program umí vytvořit soubor s uvedeným datem modifikace.

```
touch -d "2007-08-13 18:08:53" /tmp/$$.ts1
touch -d "2007-08-13 18:09:07" /tmp/$$.ts2
...
if [ /tmp/$$.ts1 -nt /tmp/$$.ts2 ]; then ... fi
...
rm /tmp/$$.*
```

V uvedeném příkladu vytváříme soubory v adresáři `/tmp` a a do jejich jména zahrnujeme číslo procesu (`$$`). Činíme tak proto, aby při běhu více programů využívající této technologie srovnání času nedocházelo ke kolizím na jménech souborů.

Dalším možným způsobem jak porovnávat data a časy je porovnávat jejich textové reprezentace. Zde z výhodou využijeme programu **date**, který dovoluje námi stanovené datum formátovat potřebným způsobem.

```
date1=$(date -d "2007-08-13 18:17:11" +%F_%T)
date2=$(date -d "2 days ago" +%F_%T)
...
if [[ $date1 < $date2 ]]; then ... fi
```

Tento způsob má oproti předchozímu dvě podstatné výhody.

- nepoužívá soubory na filesystému a je tedy rychlejší
- změnou formátovacího parametru `+%F_%T`, `+%F`, `+%T`, ... dosáhneme sofistikovanějšího porvnání například jen podle data či jen podle času. Můžeme porovnávat například i podle dne v týdnu.

60.8.2. Počítání s časem

Odkazy:

- Unix time (http://en.wikipedia.org/wiki/Unix_epoch) na Wikipedii

- date-calculation (examples/bash/date/date-calculation)

Nejlepší, alespoň tak se mi to jeví, je pro potřeby výpočtu si převést čas na počet sekund od začátku nové éry (http://en.wikipedia.org/wiki/Unix_epoch). Získáme tak obyčejné dlouhé celé číslo, se kterým se dá již pracovat. Tak tedy nejdříve to číslo

```
aktualni_sekunda=$(date +%s)
```

Nyní nějak konkrétní čas či datum v minulosti

```
okamzik_v_minulosti=$(date -d 2010-02-13 +%s)
```

Můžeme teď spočítat okamžik uprostřed

```
((okamzik_uprostred = (aktualni_sekunda - okamzik_v_minulosti)/2 + okamzik_v_minulosti))
```

Když chceme čas v sekundách převést, použijeme zápis se znakem @

```
date -d@$okamzik_uprostred +%F\ %T
```

60.9. Logické hodnoty (boolean)

Pokud si v programu deklarujeme logické hodnoty `TRUE` a `FALSE` následujícím způsobem,

```
declare -ir TRUE=1 FALSE=0
```

mají tu správnou hodnotu pro aritmetické výrazy. Platí tedy:

```
if (($TRUE)); then
   : TRUE
else
   : FALSE
fi
```

60.10. Pole proměnných

Obrázek 60-1. Pole ve zkratce

```
$ declare -a kamaradi
$ kamaradi=([1]=Karel Tomáš Pavel Hanka)
$ echo ${kamaradi[2]}  #=> Tomáš
$ echo ${kamaradi[*]}  #=> Karel Tomáš ...
$ echo ${#kamaradi[*]} #=> 4 (počet prvků)
$ echo ${!kamaradi[*]} #=> 1 2 3 4 (seznam indexů)
```

V Bashi máme k dispozici jednorozměrná pole. Pole nemusíme deklarovat, můžeme jej rovnou použít. Pokud jej chceme deklarovat, což doporučuji, můžem tak učinit příkazem **declare**.

```
declare -a name
declare -a name[subscript]
```

Obě varianty zápisu jsou ekvivalentní. Případné informace o indexu v hranatých závorkách jak je vidět na druhém řádku bash ignoruje.

```
$ declare -a mojePole
```

Přiřazení hodnoty do pole se provede jednoduše podobně jako přiřazení do proměnné.

pole[index]=hodnota

pole[*index*]=*hodnota*

Jako index můžeme použít libovolná celá čísla, tedy i čísla záporná. Velikost pole není nijak omezena. Pokud nastavíme vlastnosti pole, například jen pro čtení **readonly mojePole**, je tato vlastnost vztažena ke všem prvkům pole.

```
$ declare -a vysledky
$ vysledky[1]=dobry
$ vysledky[2]=spatny
$ vysledky[3]=pokus
bash: vysledky: readonly variable
$ vysledky[1]=vyjimecny
bash: vysledky: readonly variable
```

Do pole nemusíme zapisovat hodnoty postupně, můžeme je zapsat jedním příkazem přiřazení.

pole=([index$_1$]=hodnota$_1$ [index$_2$]=hodnota$_2$ ... [index$_n$]=hodnota$_n$)

Poznámka: Povšimněte si, že jednotlivé hodnoty jsou oddělovány mezerami a nikoliv čárkami.

Část [index$_1$]= je nepovinná. Můžeme tedy psát:

```
$ mojePole=([1]=a b c)
# nebo
$ mojePole=(a b c)
```

Přiřazení probíhá tak, že jednotlivé hodnoty v závorce jsou postupně z leva do prava přiřazovány indexům pole. Pokud není index definován, použije se index o jedničku větší než se použil u předchozí hodnoty. U první hodnoty, pokud nebyl rovněž definován, se použije index 0. Tento jednoduchý postup může přepsat právě přiřazenou hodnotu například:

```
$ mojePole=([2]=a [1]=b c)
```

hodnota c se zapíše do indexu 2, čímž přepíše hodnotu a. Příkaz je tedy ekvivalentní příkazu.

```
$ mojePole=([1]=b [2]=c)
```

Poznámka: Připomínám že toto hromadné přiřazení do pole přepíše/odstraní všechny předchozí prvky pole.

Přístup k jednotlivým prvkům pole zajistí zápis

${jméno[index]}

```
>$ mojePole=(jablko hruska svestka)
```

```
$ echo ${mojePole[2]}
svestka
```

Na místě indexu můžeme použít speciální symobly `@` a `*`. v takovém příp je výsledkem seznam všech prvků pole. Rozdíl mezi `@` a `*` je jen v případě že je celý výraz uzavřen závorkami. Rozdíl je stejný jako pří použití `$*` a `$@` proměnných.

```
$ kamaradi=(Pavel Karel Tomáš)
$ echo ${kamaradi[*]}
Pavel Karel Tomáš
```

Ze speciálních zápisů bych dále zmínil zápis který vrací délku hodnoty v prvku pole. Tedy její počet znaků.

${#pole[1]}

Obdobný zápis s použitím `*` jako indexu vrací celkový počet prvků pole.

```
$ echo ${#kamaradi[*]}
3
```

Seznam obsazených indexů pole, tedy těch čísel kterým je přiřazena hodnota získáme zápisem.

${!pole[*]}
${!pole[@]}

```
$ echo ${!kamaradi[*]}
```

60.11. Omezení programů

Základní omezení proramu můžeme provádět přikazem **ulimit**.

60.11.1. Omezení doby běhu programu

Bash sám obsahuje příkaz **ulimit** kterým se dá omezit množství procesorového času spotřebovaného programem.

```
...
(
    ulimit -t 300               # 5 minut
    program parametry
)
```

Jen připomínám že se jedná o omezení procesorového času, tedy sčasu kdy je programu přidělen procesor. Nejedná se tedy o omzení celkové doby běhu programu. Toto omezení musíme provést jinak.

```
#!/bin/bash
...
# Killing process with all it's children.

function killtree() {
    local -r parent=$1
    local -r children=$(pgrep -P $parent)
    kill -s KILL $parent
    for child in $children; do
 killtree $child
```

```
    done
} #killtree()

(
    program parametry
)&
program_pid=$!

# Spawn the guardian.  Guardian will watch the program and kill it if
# it runs too long.
(
    sleep 20m
    # If we are there, program runs out of time.  So kill it.
    killtree $program_pid
)&
guardian_pid=$!

# Wait for the program or guardian to end.
wait $program_pid
program_status=$?

if [[ $program_status == 137 ]]; then
    echo "Program killed because it runs out of time."
    wait $guardian_pid
else
    echo "Program status is $program_status."
    killtree $guardian_pid
fi
...
```

60.12. Ukončení programu včetně všech jeho potomků

V praxi jsem se setkal s potřebou ukončit proces jež pid je známo v četně všech jeho potomků. Prozkoumal jsem všechny dostupné přiakzy a zjistil že žádný z nich tuto možnost neposkytuje. Ve svých skriptech proto používám jednoduchou funkci která toto umí. Ta vychází z toho, že sice nemohu zjistit všechny potomky procesu, ale mohu zjistit všechny potomky v první generaci. Rekurzivně pak procházím potomky těchto potomků. Výsledný kód je velmi jednoduchý.

```
#!/bin/bash
...
# Killing process with all it’s children.

function killtree() {
    local -r parent=$1
    local -r children=$(pgrep -P $parent)
    kill -s KILL $parent
    for child in $children; do
 killtree $child
    done
} #killtree()
```

60.13. Zabezpečení skriptu před vícenásobným spuštěním

Odkazy:

- Lock your script (against parallel run) (http://bash-hackers.org/wiki/doku.php/howto/mutex)

U některých programů potřebujeme zajistit aby nedošlo k jejich vícenásobnému paralelnímu spuštění. Můžet to být například proto, že pracují s jedinečnými zdroji.

Dříve jsem používal pro ochranu před vícenásobným spuštěním externí program **lock**. Bohužel už si nevzpomenu na název balíčku, jehož byl tento program součástí. Protože tento program není ve standardní istalaci, hledal jsem řešení jen s pomocí samotného bash. Delší dobu jsem používal jednoduchou metodu založenou na zámkovém souboru. Tuto metodu ilustruje následující ukázka. Zámkový soubor se používá také pro uložení čísla procesu. Princip je takovýto. Jestliže existuje zámkový soubor, tak již beží jiná instance programu. Ukončíme tedy skript. Jestliže zámkový soubor neexistuje, pak žádná jiná instance programu nebeží a můžeme tedy zamknou zámek pro sebe a označit si do něj číslo aktuálního procesu. Po ukončení programu zase musíme zámek odstranit, což realizuji buď to příkazem **rm $PIDFILE** na konci programu, nebo definuji trap jako v ukázce.

```
declare -r PIDFILE=/var/run/${0##*/}

if [[ -f $PIDFILE ]]; then
    echo "${0##*/} is locked so exitting." >&2
    exit
else
    echo $$ >$PIDFILE
    trap 'rm -r $PIDFILE' 0
fi
```

Tento přístup má zásadní problém. Testování existence souboru a jeho vytváření jsou dvě operace jenž probíhají v čase po sobě. Teoreticky může nastat situace kdy je program spuštěn dvakrát ve stejném čase. Ve stejném čase budou obě instance testovat přítomnost souboru a obě projdou a budou pokračovat dál v domění, že je vše v pořádku. Velmi záleží na tom, jestli je reálná pravděpodobnost, že dojde k současnému vícenásobnému spuštění programu. Pokud ne, například když spouštíme program z cronu, není pro nás tato race condition (http://cs.wikipedia.org/wiki/Race_condition) tak nebezpečná.

Pokud nám to nestačí můžeme s výhodou použít atomické operace **mkdir $LOCKDIR**. Na tento způsob jsem narazil při pročítání Lock your script (against parallel run) (http://bash-hackers.org/wiki/doku.php/howto/mutex). Zde uvádím jen velmi zjednodušenou verzi.

```
declare -r LOCKDIR=/var/run/${0##*/}.d
declare -r PIDFILE=$LOCKDIR/pid

if mkdir $LOCKDIR; then
    echo $$ >$PIDFILE
    trap 'rm -rf $LOCKDIR' 0
else
    exit
fi
```

Další možné způsby jsou v podstatě variace na předchozí. Vždy potřebujeme kód s if kde v podmínce je atomická zamykací operace.

```
if lock; then
    # locked
```

```
else
    # Another instance running
    exit
fi
```

60.14. Extrémní programování v shellu

Odkazy a zdroje:

- XProgramming — Software Downloads (http://www.xprogramming.com/software.htm)
- ShUnit for Korn shell (http://sourceforge.net/projects/shunit/)
- xUnit Testing Framework (http://sourceforge.net/projects/xunit/)
- _ ()

Dlouhou dobu jsem psal skripty v Bashi bez jakéhokoliv systematického testování. Poté co jsem se začal zajímat o praktiky extrémního programování mi velmi začal scházet systém testů pro Bash. Když jsem tedy 2004-11-16 našel xUnit Testing Framework (http://sourceforge.net/projects/xunit/), který byl ve stádiu plánování, ale již obsahoval uvolněný balíček `bashunit` verze 1.0 **FIXME:**

Ke studiu:

- Test Anything Protocol (http://en.wikipedia.org/wiki/Test_Anything_Protocol)
- TAP (Test Anything Protocol) (http://testanything.org/wiki/index.php/Main_Page) Main Page
-
-

60.14.1. ShUnit

ShUnit je ke stažení z Source Forge (http://sourceforge.net/projects/shunit/). Poslední dostupná verze (k 2005-06-02) je verze 1.2 z 2004-12-24.

Po stažení a rozbalení balíčku jsem objevil tyto soubory:

```
radek@yoda:~/tmp/ShUnit-1.2$ ls -l
celkem 32
-rw-r--r--    1 radek    radek         415 pro 24 05:45 logTest
-rw-r--r--    1 radek    radek         632 pro 24 05:45 monitorTestAndNotify
-rw-r--r--    1 radek    radek         433 pro 24 05:45 notify
-rw-r--r--    1 radek    radek        3947 pro 24 05:45 shUnit
-rw-r--r--    1 radek    radek        1545 pro 24 05:45 shUnitPlus
-rw-r--r--    1 radek    radek        1941 pro 24 05:45 shUnitPlusTest
-rw-r--r--    1 radek    radek        1871 dub 26  2004 shUnitPlusTestFailing
-rw-r--r--    1 radek    radek         828 pro 24 05:45 shUnitTest
```

FIXME:

Použití:

1. Write a script.
2. In the 'main' of the script, the shUnit functions have to be sourced in (. `/wherever/the/shUnit/file/is/shUnit`)
3. Create a function in your test scripts that prepares the environment in which your test will be run
4. Again in the `main`, call the `shuStart` function with the name of the initia lization function as an argument.
5. For each testfunction, perform these steps:

5.1. Add a line to the initialization function to register the testfunction. This will look like: **shuRegTest NameOfTestFunction**
5.2. Create a function `NameOfTestFuction` that
1. executes whatever you wish to test
2. Checks the result of what you've just tested
3. Calls the shuAssert function to assert that the result is as expected
4. You can put as many asserts as you wish in each testfunction.

6. Run your testscript

1. Write a script.
2. In the 'main' of the script, the shUnit functions have to be sourced in (`. /wherever/the/shUnit/file/is/shUnit`)
3. Create a function in your test scripts that prepares the environment in which your test will be run
4. Again in the `main`, call the `shuStart` function with the name of the initialization function as an argument.
5. For each testfunction, perform these steps:
5.1. Add a line to the initialization function to register the testfunction. This will look like: `shuRegTest NameOfTestFuncti`
5.2. Create a function `NameOfTestFuction` that
5.2.1. executes whatever you wish to test
5.2.2. Checks the result of what you've just tested
5.2.3. Calls the `shuAssert` function to assert that the result is as expected
5.2.4. You can put as many asserts as you wish in each testfunction.
6. Run your testscript

You can of course repeat step 5 as much as you like (many testfunctions).

60.14.1.1. Analýza skriptů

Podíváme se do jednotlivých souborů a popíšeme si funkce v nich definované.

60.14.1.1.1. `shUnit`

Vlastní `shUnit`, původně od ...

`shuRegTest`(*`strFunction`*)

Registrování testovací funkce/metody.

shuRunOneTest(strTestToRun)

FIXME:

StrToPrint(sHowMany)

FIXME:

shuStart(strInitFunction)

FIXME:

shuAssert(strMessage, boolResult)

FIXME:

shuDeny(strMessage, boolResult)

FIXME:

60.14.1.1.2. `shUnitTest`

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

```
. ./shUnit


TestCase1()
```

FIXME:

```
    TestCase1() {
        shuAssert "Test1" ${TRUE}
        shuDeny "Test1" ${FALSE}
    }
```

```
TestCase2()
```

FIXME:

```
    TestCase2() {
        shuAssert "Intentionally broken" ${FALSE}"
    }
```

```
TestCase3()
```

FIXME:

```
    TestCase3() {
        [ 7 -eq 7 ]
        shuAssert "Test4: 7 equals 7?" $?
        [ 5 -eq 5 ]
        shuAssert "Test4: 5 equals 5?" $?
    }
```

```
TestCase4()
```

FIXME:

```
    TestCase4() {
        [ 7 -eq 6 ]
        shuAssert "Intentionally broken: 7 != 6" $?
    }
```

```
InitFunction()
```

FIXME:

```
    InitFunction() {
        shuRegTest TestCase1
        shuRegTest TestCase2
        shuRegTest TestCase3
        shuRegTest TestCase4
    }

### Main
shuStart InitFunction
```

60.14.2. shunit2

Odkazy:
- shunit2 (http://code.google.com/p/shunit2/) na Google Code
-

60.14.3. Bashunit

Odkazy:
- bashunit (http://ostatic.com/bashunit) na OSTATIC
- na freshmeat (http://freshmeat.net/projects/bashunit)
- xUnit - Unit Testing Framework (http://sourceforge.net/projects/xunit/) na Source Forge
-

FIXME:

60.14.4. testsimple

Odkazy:
- testsimple (http://code.google.com/p/testsimple/) na Google Code
- Test.Simple (http://openjsan.org/doc/t/th/theory/Test/Simple/) od David Wheeler
-
-

60.15. Obrana před vlastními chybami

Odkazy:
- Writting Robust Shell Scripts (http://www.davidpashley.com/articles/writing-robust-shell-scripts.html) na DavidPashley.com
-
-
-

Člověk je tvor chybující, a programátor/administrátor by si to měl uvědomovat dvakrát. Jeho největším nepřítelem jsou vlastní chyby. Abych nezůstal jen v obecné rovině, ukažme si příklad.

```
dir=$1
...
rm -fr $dir/bin
```

Na první pohled je patrné, co se ve skriptu děje. Mažeme podadresář bin zadaného adresáře jenž byl předán jako první parametr ve skriptu. Jasné, pochopitelné. A nyní si představme, a opravdu to nezkoušejme, že uživatel skriptu zapomene zadat parametry skriptu. Proměnná `dir` bude prázdná a příkaz pro mazání se provede jako **rm -fr /bin**. Tohle jsme ale vůbec nezamýšleli. Pokud skript spustí administrátor, a zapomene zadat parametry, pohroma je na světě. Podobných nešťatných chyb, které vlastně nejsou chybami v pravém slova smyslu, je možno udělat více. Nyní si ukážeme různé techniky jak podobným situacím předejít.

Nejdříve se podíváme na možnosti nastavení bashe. Máme řadu přepínačů na příkazové řádce kterými ovlivníme chování bashe. Tyto přepínače/volby se dají nastavit i programově, "zevnitř" programu, pomocí příkazu **set**.

Nastavení **set -u** (**set -o nounset**), způsobí že bash považuje za chybu použití jakékoliv proměnné, jejíž obsah nebyl před tímto použitím definován. Tedy proměnné jenž nebyla "inicalizována".

Nastavení **set -e** (**set -o errexit**), způsobí ukončení vykonávání programu pokud libovolný příkaz skončí s chybou. Toto nastavení, tedy tato změna chování bashe, vyžaduje změnu přístupu v programování. Již nemůžeme použít něco takovéhoto:

```
příkaz
if [[ $? != 0 ]]; then
 echo "Příkaz neuspěl"
        # ošetření chyby
fi
```

Pokud se chceme dostat k návratové hodnotě příkazu, využjeme možnosti spojení příkazů operátorem `||`.

```
#!/bin/bash
set -o errexit
cd /nikam || echo "\$?=$?"
```

60.16. Řešení

Řešení různých situací a potřeb.

60.16.1. HotFolder

HotFolder je adresář, který po vložení souborů tyto zpracuje a předá jinam. Je to jeden ze stavebních kamenů produkčního dokumentového řetězce. Jednoduchým příkladem je produkční řetězec pro zpracování obrázků. Po přesunutí/nakopírování obrázku do adresáře je tent upraven/zmenšen/zvětšen/... a přemístněn do jiného, výstupního adresáře.

Technicky HotFolder řeším skriptem, který je opakovaně spouštěn z cronu a testuje zdali jsou v zadaném adresáři přítomny nějaké soubory.

60.16.1.1. Diskuse řešení

Protože hotfolder musí kontrolovat adresář, existuje jen několik cest jak toho dosáhnout.

Prvním způsobem ja periodicky kontrolovat stav adresáře a podle zjištěných změn se rozhodnout který soubor zpracovat. Periodicita je zajištěna buď'to spouštěním programu přes cron, nebo program funguje v nekonečné smyčce.

```
# Endless loop
while true; do
        process_hotfolder
        sleep $INTERVAL
done
```

Pokud spouštíme program z cronu, musíme vyřešit zámek. Zámek který zajistí aby se program nespustil vícekrát, pokud z takového konkurenčního spuštění hrozí nějaké nebezpečí.

```
if lockfile -r0 $LOCK; then
        process_hotfolder
        rm -f $LOCK
else
        echo "HotFolder locked"
fi
```

Dalším úkolem je vyřešit rozpoznávání souborů. Myslím tím ten správný okamžik kdy již můžeme soubor začít zpracovávat. Musíme mít na paměti, že pokud povolíme i jiný způsob umístnění souboru do adresáře než je příkazem **mv**, tedy způsob například příkazem **cp**. Druhý způsob může být znázorněn také programem který přímo zapisuje soubor do adresáře. Takový program může běžet delší dobu, například i několik minut.

Můžeme:

- sledovat čas modifikace souboru
- sledovat velikost souboru
- sledovat kontrolní součet souboru
- kombinovat výše uvedené způsoby

Po úvaze jsem zvolil první způsob. Tedy sledování času modifikace souboru. Tento způsob pro všechna nasazení které jsem zatím dělal plně vyhovoval. Princip je takový, že periodicky zkoumám obsah adresáře příkazem **find** a zajímám se o soubory jejich čas modifikace je starší než `$SURETIME`.

```
soubory=$(find $HOTFOLDER -mmin +$SURETIME)
```

Při tomto způsobu mohou být soubory zpracovány v jiném pořadí než v jakém byly do hotfolderu vloženy. Opět, ve věech případech kdy jsem hotfolder použil to nebyl problém. Nicméně i tato záležitost jde řešit. Například takto

```
soubor=$(ls -tr|head -n1)
if [[ -n "$soubor" ]]; then
    if [[ $(date +%F_%T -r $soubor) < $(date +%F_%T -d "3 mins ago") ]]; then
        zpracuj_soubor $soubor
    fi
fi
```

Kreativitě se meze nekladou a možností je hodně.

Pokud není možné využít časové značky souboru, například proto že nemáme program jenž soubory umístňuje do adresáře pod úplnou kontrolou. Tento modifukuje časové značky způsobem jenž je pro nás velmi nešťastným. V takovém případě nezbyde než si detekci vložení souboru naprogramovat jiným způsobem. Například hlídáním kontrolních součtů souborů. Takové řešení ale vyžaduje pamatovat si mezi jednotlivými spuštěními našeho programu stav kontrolních součtů souborů a ukládat si jej. Můžeme mít databázi s takovouto strukturou.

soubor:čas:md5sum

Adresář pak procházíme a pro každý soubor v něm se rozhodneme podle souboru a jeho záznamu v databázi jestli záznam v databázi poopravíme a soubor zatím nezpracujeme. Nebo můžem soubor začít zpracovávat.

60.17. Různé nezapracované texty a poznámky

there is a new \D{...} prompt expansion; passes the '...' to strftime(3) and inserts the result into the expanded prompt

quoting řetězců `$'...'` a `$"..."`

variables: BASH, BASH_VERSION, BASH_VERSINFO, UID, EUID, REPLY, TIMEFORMAT, PPID, PWD, OLDPWD, SHLVL, RANDOM, SECONDS, LINENO, HISTCMD, HOSTTYPE, OSTYPE, MACHTYPE, HOSTNAME, ENV, PS3, PS4, DIRSTACK, PIPESTATUS, HISTSIZE, HISTFILE, HISTFILESIZE, HISTCONTROL, HISTIGNORE, GLOBIGNORE, GROUPS, PROMPT_COMMAND, FCEDIT, FIGNORE, IGNOREEOF, INPUTRC, SHELLOPTS, OPTERR, HOSTFILE, TMOUT, FUNCNAME, histchars, auto_resume

DEBUG trap, ERR trap

přesměrování: <>, &>, >|, <<<, [n]<&word-, [n]>&word-

Ukázka čtení z databáze

```
rm -f pipe
mkfifo pipe

mysql $CREDENTIALS -e "SELECT id,name FROM stoly;" >pipe &

exec 3<pipe
while read -u3 id name rest; do
        echo "$id, rest=$rest"
done
exec 3<&-
```

60.18. Úvahy a náměty

60.18.1. OOP

Poznámky a náměty k realizaci OOP v bashi.

2007-02-15

Pokus o konstrukci jednoduchého oběktově orioentovaného systému v bashi.

Začal jsem uvažovat o objektově orientovaném programování v bashi. Na mysli mi přišla představa kterou tady popíši a kterou chci vyzkoušet.

Objektově orientované programování, dál již jen OOP má několik základních vlastností, jsou to:

- Zapouzdření (Encapsulation)
- Polymorfismus

Zapouzdření (Encapsulation) znamená že existuje hranice ktaré vyděluje vnitřní části objektu od okolního světa.

Polymorfismus, to je, že každý objekt reaguje na stejnou zprávu mu zaslanou z vnějšku svým vlastním osobitým způsbem. Mějme dva objekty ”a” a ”b”. Každému z nich pošleme stejnou zprávu

```
# a pozdrav
dobrý den
# b pozdrav
čau
```

Tedy oba objekty dostali stejnou zprávu ale každý na ni reaguje jiným způsobem.

Objekty. V bashi nemáme mnoho možností jak reprezentovat objekty, paměťové struktury jsou značně primitivní. Zbývá použít souborový systém. Objekt tedy reprezentujeme nějakým elementem soborového systému. K dispozici máme jen dvě možnosti: soubor a adresář.

Objekt reprezentovaný souborem. Pokud budeme chtít reprezentovat objekt souborem, tedy jedním souborem, musíme se vyrovnat s tím, jak do něj uložit vše potřebné, tedy metody a atributy. T.j. v souboru budou části reprezentující jednotlivé části oběktu a při měnách v těchto částech musí být soubor měněn (editován). Editován myslím progamově.

Objekt si můžeme představit jako skript/program. Volání metody objektu je pak spuštění programu s metodou jako parametrem:

```
# obj method
```

```
# obj method
```

Celkově se mi zdá tento způsob implementace objektů nevhodný z důvodů komplikovného meodfikování atributů objektu.

Objekt reprezentovaný adresářem. Druhá možnost je reprezentovat objekt skupinou souborů, každý pro jeden atribut/metodu. Změna hodnoty atributu se pak uskuteční jednoduše přepsáním souboru. Například nastavení atributu **var** na hodnotu 45 realizujeme snadno následovně:

```
echo 45 >var
```

Soubory můžeme "svázat" jménem, respektive jeho částí, nebo adresářem. V tomto případě tvoří obsah adresáře jeden objekt.

Máme-li jméno objektu (adresáře) v proměnné **obj** , můžme volat metodu **metoda** s parametrem 42 takto:

```
$obj/metoda 42
```

Zprávu objektu pak pošleme pomocí zprostředkovatelské funkce/příkazu **sendmsg**:

```
sendmsg $obj metoda 42
```

Základní metody:

```
obj addSlot: <symbol>
obj addSlot: <symbol> valued: <value>
obj addImmutableSlot: <symbol> valued: <value>
obj removeSlot: <symbol>
```

Objekty jsou reprezentovány adresáři. Jeden adresář jeden objekt. Uvnitř adresáře jsou soubory reprezentující jednotlivé části objektu jako jsou:

- metody
- atributy

60.18.1.1. Varianta první

V této variantě nepoužívám třídy, jso prostě jen objekty. Nový objekt nevzniká tedy tak že by jej vytvořila třída. Nový objekt vzniká jako kopie jiného objektu.

60.19. Staré texty

FIXME: TBD:Zapracovat do předchozích sekcí.

60.19.1. Začínáme

Předtím než přikročíme k samotnému programování/skriptování musíme si projít řadu konceptů jenž jsou nezbytné k pochopení.

60.19.1.1. Příkaz

Příkaz je externí (z hlediska shellu) program, vestavěný příkaz nebo definovaná funkce. Má své jméno kterým se volá, předávají se mu parametry (píší se za jméno), a má standardní I/O proudy (vstupní, výstupní a chybový).

Omezení kladená na jméno příkazu jsou určená tím je-li příkaz externím programem, pak jsou stejná jako omezení na jméno souboru. Je-li příkaz funkcí sestává jeho jméno jen z písmen, číslic a znaku „_“.

60.19.1.2. Přesměrování I/O proudů

cmd < vstupní-soubor > výstupní-soubor 2> záznam-chyb

V bashi máme mechanismus jak přesměrovat jednotlivé I/O proudy do souborů, nebo při čtení, ze souborů slouží k tomu znaky „<“ a „>“. První z nich přesměrovává vstupní proudy, druhý pak výstupní.

cmd >> výstupní-soubor

60.19.1.3. Kolony

FIXME:

$ **prog1 | prog2 | prog3 | prog4**

60.19.1.4. Spojování příkazů

prog1; prog2; prog3;

```
$ prog1 && prog2
```

```
$ prog1 || prog2
```

60.19.1.5. Seskupování příkazů

FIXME:

```
$ (prog1; prog2)
```

60.19.2. Příkazy

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

ToDo

1. Programy, vyhledávání programů.
2. Standardní vstup, výstup, chybový výstup
3. Řazení příkazů
4. Kolony

FIXME:

60.19.3. Proměnné

ToDo

1. Nastavení proměnné
2. Použití proměnné
3. Expanze

FIXME: Expanze

```
$name
${name}
${name##maska}
${name%maska}
${name%%maska}
${name/vzor/náhrada} -- vymění první výskyt vzor za náhrada
${name//vzor/náhrada} -- vyměný všechny výskyty vzor za náhrada
⋮
```

```
$var
${var}
```

Vloží obsah proměnné na místo výskytu zápisu `$var` nebo `${var}`. Oba zápisy jsou ekvivalentní. Druhý zápis je přesnější a použijeme jej s výhodnou například v případě `${var}_name`. Znak „_“ je totiž legálním znakem identifikátoru a bez složených závorek `{}` by shell hledal proměnnou `$var_name`.

${var}

FIXME:

${var:*n*}

Vrátí text uložený v proměnné od znaku *n* do konce.

${var:*s*:*l*}

Text uložený v proměnné od pozice *s* v délce *l* znaků.

${#var}

Délka textu uloženého v proměnné, počet znaků.

${var#*maska*}

FIXME:

```
$ var=soubor.txt
$ echo ${var#*.}
txt
```

$

FIXME:

$

FIXME:

${var:-word}

FIXME:

${var:=word}

FIXME:

${var:?word}

FIXME:

${var:+word}

FIXME:

$

FIXME:

$

FIXME:

60.19.4. Řídící struktury

60.19.4.1. Podmíněné větvení skriptu

Základní konstrukcí je podmíněné vykonání příkazu či bloku příkazů. Jeho struktura vypadá následovně:

```
if program
then
  příkazy
fi
```

Kde program zde figuruje jako podmínka. Je spuštěn a na základě jeho návratové hodnot jsou či nejsou vykonány příkazy mezi **then** a **fi**.

* ***FIXME:*** *Popsat význam návratových kódů.*

Program uvedaný v části podmínky příkazu **if** se spustí a jeho návratová hodnota určuje která větev se provede. Návratové hodnoty programu v roli podmínky jsou:

- `0` — true, pravda
- `1` až `255` — `false`, nepravda

```
if condition
then
    command
elif condition
then
    command
else
    defualt-command
fi
```

```
if [ -f soubor ]; then
    ...
else
    ...
fi
```

```
case v in
    wr)
        ...
        ;;
    *)
        ...
        ;;
esac
```

Dále uvedu jednoduché příkalady, ukázky použití.

Máme hodnotu v proměnné v jenž prezentuje pravdivostní hodnotu 0-true / 1-false.

```
v=0
if ((v)); then
    # Tato část se provede jen pro v != 0
else
    # Tato část se provede jen pro v == 0
fi
```

60.19.4.2. Cykly

Nekonečný cyklus.

```
while :
do
    ...
done
```

```
for arg in list
do
    command
done
```

```
while condition
```

```
do
    command
done
```

```
until condition
do
    command
done
```

60.19.5. Aritmetika v bashi

Aneb počítáme s bashem

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

2 + 2 = 5, for extremely large values of 2.

unknown

Krátky přehled způsobů počítání

- z=$((z+3))
- ((z=z+3)) ((z+=3))
- **let** z=z+3
- **let** z+=3
- **let** "z += 3"
- $[výraz]
- ((výraz))
- .
- .
- eval výraz
- **expr** — externí program
- Použitím dalšího externího programu jako je **dc**, **bc**, **awk**, **perl**, **python**, **ruby** či jiný.

V bashi můžeme také provádět jednoduché výpočty.

Vyhodnocování matematických výrazů leze uskutečnit několika způsoby. Prvním z nich je použití funkce `eval`.

60.19.5.1. expr

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Externí program **expr** nám můž s výhodou posloužit pro realizaci výpočtů. Mimo práci s čísly ma zabudovány i jednoduché operace s řetězci.

```
$ v=$(expr 3 \* 2)
$ echo $v
6
```

60.19.6. Zabudované proměnné

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

`$0`

Název spouštěného programu/skriptu. Jedná se o úplný název včetně celé cesty k souboru s právě spuštěným skriptem.

`$1 $2 ... ${n}`

Poziční parametry skriptu/programu či funkce. Prvních devět parametrů je přístupno přes zápis `$n` i `${n}`. Další parametry pak z pochopitelnýc důvodů jen přes `${n}`. (Příklad `$12` je `${1}2`)

`$*`

Všechny parametry reprezentované jako jeden řetězec. Tedy `$1 $2 ... ${n}`.

`$@`

Za běžných okolností funguje stejně jako proměnná ***`$*`***. Její chování se líší je-li oraničena uvozovkami, tedy v zápisu `"$@"`. V tom případě je nahrazena seznamem argumentů `"$1" "$2" "$3" ... "${n}"` narozdíl od proměnné ***`$*`*** jenž je v takovém zápisu nahrazena `"$1 $2 $3 ... ${n}"`.

`$#`

Počet parametrů. Tedy počet parametrů předaných skriptu/programu, nebo počet parametrů předaných do funkce.

`$?` — exit status variable

Proměnná obsahuje status posledně vykonaného příkazu. Tedy návratovou hodnotu tohoto příkazu/programu/skriptu.

`$-`

Aktuální příznaky/přepínače zadané při spuštění, nastavené příkazem **set** nebo implicitní pro bash.

`$$` — process id variable

Pid procesu, aktuálně běžícího. V subshelu spuštěném v `( )` vrátí pid volajícího shellu nikoli subshelu. **FIXME:**ověřit.

`$!`

Pid posledně spuštěného procesu na pozadí. Tedy posledního příkazu či skupiny příkazů spuštěné asynchronně, t.j. zakončené znakem `&`.

`$_`

FIXME:

`$BASH`

Cesta k samotnému programu bash.

`$BASH_ENV`

FIXME:

`$BASH_VERSINFO[n]`

Šestiprvkové pole obsahující informaci o instalované (běžící) versi bashe. Význam jednotlivých prvků pole je následující:

- `BASH_VERSINFO[0]` — hlavní (major) číslo verze (RELEASE).
- `BASH_VERSINFO[1]` — vedlejší (minor) číslo verze (VERSION).
- `BASH_VERSINFO[2]` — verze záplaty (*patch level*)
- `BASH_VERSINFO[3]` — verze sestavení (*build level*)
- `BASH_VERSINFO[4]` — status vypuštění (*release status*) (např. BETA1)
- `BASH_VERSINFO[5]` — hodnota MACHTYPE

`$BASH_VERSION`

Informace o verzi bashe. Obsah této proměnné je seskládán z prvků pole `BASH_VERSINFO`

`$DIRSTACK`

FIXME:

`$EDITOR`

Implicitní editor volený skripty. Obvykle **vi** nebo **emacs**.

`$EUID`

„Efektivní“ id uživatele.

`$FUNCNAME`

Jméno aktuální funkce.

```
fce23 ()
{
    echo "Running: $FUNCNAME($*)"
}
```

`$GLOBIGNORE`

FIXME:

`$GROUPS`

Skupiny ke kterým přináleží aktuální uživatel pod nímž skript běží.

`$HOME`

Domovský adresář uživatele.

`$HOSTNAME`

Jméno stroje.

`$HOSTTYPE`

Typ stroje.

`$IFS`

Oddělovač polí na vstupu.

`$IGNOREEOF`

FIXME:

`$LANG`
`$LC_COLLATE`
`$LC_CTYPE`
`$LC_MESSAGES`
`$LC_NUMERIC`

FIXME:

`$LINENO`

Číslo řádku skriptu na kterém se tato proměnná nachází.

`$MACHTYPE`

Typ stroje.

`$MAILCHECK`

Typ stroje.

`$OLDPWD`

FIXME:

`$OSTYPE`

FIXME:

`$PATH`

FIXME:

`$PIPESTATUS`

FIXME:

`$PPID`

FIXME:

`$PS1`

FIXME:

`$RANDOM`

Slouží ke generování náhodných čísel. Pokaždé když je tato proměnná použita, je vybráno náhodně číslo z rozsahu 0 až 32767. Zápis do této proměnné inicializuje generátor náhodných čísel.

`$SECONDS`

Tato proměnná obsahuje počet sekund které uběhly od okamžiku spuštění shellu. Zápis do této proměnné ji nastaví na definovanou hodnotu. Poté zobrazuje počet uběhlých sekund od okamžiku zápisu plus zapsanou hodnotu.

`$TIMEFORMAT`

FIXME:

`$TMOUT`

Doba nečinnosti v sekundách po které se bash ukončí.

`$UID`

Id číslo uživatele. Proměnná je jen pro čtení.

60.19.7. Argumenty skriptu

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Odkazy:

- Handling Command Line Arguments (http://www.shelldorado.com/goodcoding/cmdargs.html)

Náš skript, jako každý program, má přístup k parametrům uvedeným na příkazovém řádku. Zpracovávat je můžeme několika způsoby, ale přístup je v základě přes pořadové číslo parametru.

$0 $1 … $9
${0} ${1} … ${23} …

Další informace které máme je celkový počet argumentů `$#` a proměnné obsahující všechny argumenty `$*` a `"$@"`

```
vflag=off
while [ $# -gt 0 ]; do
    case "$1" in
        -v)  vflag=on;;
    esac
    shift
done
```

```
vflag=off
while [ $# -gt 0 ]; do
    case "$1" in
        -v)  vflag=on;;
        -*)
            echo "usage: $0 [-v] [file ...]" >&2
            exit 1;;
        *)  break;;
    esac
    shift
done
```

```
vflag=off
filename=
while [ $# -gt 0 ]; do
    case "$1" in
        -v) vflag=on;
        -f) filename="$2"; shift;;
        ⋮
    esac
    shift
done
```

getopts optstring name [args]

60.19.7.1. getopt

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

V analýze parametrů nám může být nápomocen externí program **getopt**. Tomuto předáme parametry a on je přeskupí a uspořádá tak, aby se nám dobře a jednoduše četly.

```
$ getopt -alneco,nic a:b:v: -a1 -v2 soubor -bc -nic

set -- $(getopt f:vl "$@")
```

60.19.7.2. getopts

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

getopts je zabudovaný/vestavěný příkaz bashe. Je to shellovská varianta C funkce `getopt`.

```
vflag=off
filename=
while getopts vf: opt; do
    case "$opt" in
        v)  vflag=on;;
        f)  filename="$OPTARG";;
        \?) # unknown flag
            echo "usage: $0 ..." >&2
            exit 1;;
    esac
done
shift $(expr $OPTIND - 1)
```

FIXME:

60.19.8. Příkazy

* *rcsinfo="$Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

60.19.8.1. echo

* *section id="bash.command.echo"*
* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Příkaz **echo** slouží k výstup informací. Parametry tohoto příkazu popisují co a jak se má vypsat na standardní výstup. Samozřejmě že s použitím operátorů přesměrování dosáhneme výstupu i jiným kanálem/proudem. Často se používá také výstup do souboru. Ale niní již k samotnému příkazu **echo**.

echo [-neE] [arg . . .]

Jednotlivé argumenty *`arg`* jsou vypsány na standardní výstup odděleny mezerou „ “. Přepínače ovlivňují funkci příkazu **echo** takto:

`-n`

Potlačí znak nového řádku. Bez tohoto přepínače příkaz **echo** odešle po vypsání všech parametrů na výstup znak konce řádku. Potřebujeme-li vypisovat dalším příkazem na stejný řádek, použijeme tento přepínač.

`-e`

FIXME:

`-E`

FIXME:

60.19.9. Přehled konstrukcí

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Tato část slouží jako malý přehled programových konstrukcí.

```
[ ... ]   — vyhodnocení podmínky,  jiný zápis pro příkaz test

$var            -- použití proměnné
${var}           -- použití proměnné

expr výraz
$[ arit.výraz ]
$( program )

(( EXPRESSION ))       -- aritmetický výraz.  je ekvivalentní příkazu let
let "EXPRESSION"
$(( EXPRESSION ))

[[ EXPRESSION ]]       -- vrací 0 nebo 1

if TEST-COMMANDS; then CONSEQUENT-COMMANDS; fi
if TEST-COMMANDS; then CONSEQUENT-COMMANDS; else příkazy; fi
if TEST; then COMMANDS; elif TEST; then COMMANDS; else COMMANDS; fi

case WORD in
  PATTERN) COMMAND-LIST ;;
  ...
esac

select NAME in WORDS; do COMMANDS; done



# Cykly
for NAME in WORDS; do COMMANDS; done
for (( EXPR1 ; EXPR2 ; EXPR3 )); do COMMANDS; done
until TEST-COMMANDS; do CONSEQUENT-COMMANDS; done
while TEST-COMMANDS; do CONSEQUENT-COMMANDS; done
```

```
# v těle vcyklu mohou být užity přikazy
break          -- přeruší cyklus a opustí ho
continue       --

function fce() { příkazy }
function fce { příkazy }
fce() { příkazy }

select opt in volby; do příkazy; done    -- Jednoduché menu, interakce s uživatelem
```

60.19.10. Datové formáty a jejich čtení/zpracování

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Zdroje a odkazy:

- Data File Metaformats (http://www.catb.org/~esr/writings/taoup/html/ch05s02.html)
- .
- .

Popíšeme si a na ukázkách předvedem zpracování textových souborů jenž se na UNIXu běžně používají.

60.19.10.1. DSV *Delimiter-Separated Values*

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

V tomto typu souboru jsou jednotlivé hodnoty odděleny oddělovačem *delimiter*. Příkladem takového souboru jsou například soubory

```
/etc/passwd
/etc/shadow
/etc/group
/etc/inittab
```

Výběr pole, například si vybereme pole `gid` v soubor `/etc/passwd`

```
$ cut -d: -f4 /etc/passwd
```

60.19.10.2. Shell script formát

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Tento formát je vlstně vykonatelný shell script jehož úkolem je naplnit proměnné. Je vhodný pro konfigurační soubory.

```
# Komentář
HODNOTA=5
CYKLUS=23
SLOVO=ahoj
TEXT="Trochu delší text"
```

60.19.10.3. RFC 822 formát

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

60.19.10.4. Cookie-Jar formát

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Příklad:

```
"Among the many misdeeds of British rule in India, history will look
upon the Act depriving a whole nation of arms as the blackest."
        -- Mohandas Gandhi, "An Autobiography", pg 446
%
The people of the various provinces are strictly forbidden to have
in their possession any swords, short swords, bows, spears, firearms,
or other types of arms. The possession of unnecessary implements
makes difficult the collection of taxes and dues and tends to foment
uprisings.
        -- Toyotomi Hideyoshi, dictator of Japan, August 1588
%
"One of the ordinary modes, by which tyrants accomplish their
purposes without resistance, is, by disarming the people, and making
it an offense to keep arms."
        -- Supreme Court Justice Joseph Story, 1840
```

The cookie-jar separator was originally %%\n. I wanted something a bit more visible than % would have been. In fact, any stuff a
ter the %% is treated as a comment (or at least that's how I wrote it).
-- Ken Arnold

60.19.10.5. Record-Jar formát

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Planet: Mercury Orbital-Radius: 57,910,000 km Diameter: 4,880 km Mass: 3.30e23 kg %% Planet: Venus Orbital-Radius: 108,200,000 km Diameter: 12,103.6 km Mass: 4.869e24 kg %% Planet: Earth Orbital-Radius: 149,600,000 km Diameter: 12,756.3 km Mass: 5.972e24 kg Moons: Luna

60.19.10.6. XML

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

60.19.10.7. PYX formát

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME: Pyxie

60.19.10.8. Windows INI formát

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

60.19.10.9. Konvence pro textové soubory

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

- Je li to možno, jeden record na řádek.
- Je li to možno, řádky kratší než 80 znaků.
- Používat znak # pro komentáře.
- Podporovat konvenci obráceného lomítka. \n \r \t \b \f \e \nnn \onnn \0nnn \xnn \dnnn \\ \unnnn
- U jednořádkových recordů používat dvojtečku : nebo bílé mezery jako oddělovače polí.
- Nedělat rozdíly mezi tabelatorem a bílými mezerami.

60.19.11. Příklady

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

V této části si probereme řešení skutečných problémů se kterými jsem se potkal.

60.19.11.1. Slévání adresářů

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

Nejdříve zadání. **FIXME:** dopsat zadání. Dále upřesnění zadání:

Marvin: Takže nejdřív k zadání. Je třeba upřesnit část věcí.
Frakor: ptej se
Marvin: Kolik bude zdrojový adreářovách stromů. Málo nebo moc?
Marvin: Budou se měnit často nebo naopak vůbec.
Marvin: T.j. Bude jejich seznam přimo ve skriptu neb je třeba ho načítat z konfiguračního souboru. Nebo ho chceš předávat jako parametr skriptu.
Frakor: vzdy to bude ve tvaru .../adresar_vychozi/...vnorene_adresare
Marvin: ?, Takž se jedná jen o jediný adresářový strom?
Frakor: ano vzdy pro dany vychozi adresar
Frakor: myslel jsem ze by se to dalo predat jako input data myslim ta cesta k vychozimu adresari pro prohledavani
Marvin: jj, takže to budem ořezávat.
Frakor: tj spustim skript a on se zepta odkud ches prohledavat
Marvin: ?? nemám rád skripty co se ptají, nedají se automatizovat.
Frakor: jak mu ale specifikuji tu cestu ?
Marvin: cílový adresář, bude se měnit nebo bude pořád stejný
Frakor: cilovy se stanovi takze by mel obsahovat jmeno vychoziho adresare aby se vedelo odkud to tam je

Výsledné řešení:

Příklad 60-1. Kopírování souborů

```
#!/bin/sh
# $Header: /home/radek/cvs/unix-book/example/bash/kopirovani-souboru.sh,v 1.1.1.1 2009-01-24 1
# P<65533>enos soubor<65533>


# Pojmenov<65533>n<65533> parametr<65533>
source_dir="$1"
target_dir="$2"
str=$3

### MAIN
mkdir -p $target_dir
set -x
# Pr<65533>chod adres<65533><65533>ov<65533>m stromem
find $source_dir -iname "*$str*" -type f -exec ./kopiruj-soubor.sh "{}" $target_dir \;
```

60.19.12. Pomocné programy

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME:

60.19.12.1. Vytváření jednoduchých menu

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

FIXME: Pro vytváření jednoduchých menu používaných z shell scriptů je k dispozici řada prográmků/nástrojů. Zde několik zmíním a některé z nich dále rozvedu. Pro vytváření jednoduchých menu na terminálu/konzoli je možno použít balíčky:

- dialog
- pdmenu

Další balíčky jsou pro tvorbu a používání menu, tentokráte pod grafickou X konzolou:

- **FIXME:**tcl/tk

*60.19.12.1.1. **dialog***

* *section id="dialog" xreflabel="dialog"*
* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Program se nachází v balíčku stejného jména, takže instalace vypadá následovně:

```
# apt-get install dialog
```

V balíčku jsou mimo samotný program **dialog** příklady. Ty jsou uloženy v adresáři `/usr/share/doc/dialog/examples/`.

60.19.13. Programování CGI skriptů v bashi

* *section id="bash.cgi"*
* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Zdroje a odkazy:

- CGI Scripting (http://www.netspade.com/articles/cgi/)
- CGI Environment Variables (http://www.netspade.com/articles/cgi/variables.xml)
- Darren's Notebook: `bashlib` (http://sevenroot.org/dlc/2000/12/bashlib)
- `bashlib` - CGI programming with the bash shell (http://bashlib.sourceforge.net/)
- ----------
- CGI Programming 101: Learn CGI Today! (http://www.cgi101.com/book/)
- **FIXME:**
- **FIXME:**

ToDo list

1. **FIXME:**

Jak napovídá název, CGI skripty se dají programovat v bashi. Ostatně CGI je samo definováno tak, že vstupy (hodnoty polí, parametry předané v url a další jsou předávány v proměnných prostředí a čteny ze standardního vstupu `stdin`. Pokud tedy potřebujeme velmi jednoduché stránky, můžeme práve shell (bash) snadno použít. O tom, a taky jak daleko se dá zajít je tato část.

60.19.13.1. Část první, CGI

Než se dobereme věcí zajímavější je třeba probrat základy. Takže co je to CGI. CGI neboli *Common Gateway Interface* je rozhraní mezi webovým serverem a CGI aplikací. Toto rozhraní je definováno na úrovni prostředí. Zkrátka webový server analyzuje přišedší dotaz od prohlížeče, naplní proměnné prostředí a spustí CGI program (skript). Tento program pak na standardní výstup vypíše html kód stránky který webový server vrátí dotazujícímu se prohlížeči. Toto jednoduché rozhraní má jednu velkou výhodu. CGI program není součástí webového serveru a může být naprogramován v jakémkoliv jazyce. Může to být binární výkonný program (binárka) nebo to může být skript v libovolném skriptovacím jazyce. Zkrátka nejsme ve svém výběru vývojového prostředí žádným způsobem omezeni webovým serverem.

Než si ukážeme jednoduchý skript, připravíme si webový server, v mém případě `apache-ssl`. Do jeho konfigurace `/etc/apache-ssl/httpd.conf` vložíme buď to přímo nebo přes direktivu **Include** následující řádky.

Příklad 60-2. Konfigurace Apache

```
# Konfigurace pro na<65533>e experiment<65533>ln<65533> CGI skripty v bashi
ScriptAlias /bash/cgi-bin \
            /home/radek/document/book/programming/example/bash/cgi/bin
<Location /bash/cgi-bin>
 # Omezen<65533> p<65533><65533>stupu na u<65533>ivatele se znalost<65533> hesla
 AuthType Basic
 AuthName "Proka<65533>te se pros<65533>m heslem."
 AuthUserFile /etc/apache-ssl/passwd/user
 Require  user radek

 # Omezen<65533> p<65533><65533>stupu podle adresy u<65533>ivatele
 Order Deny,Allow
 Allow from localhost
 Deny from All
</Location>;
```

Řádek **ScriptAlias** definuje kde bude vystaven adresář s CGI skripty a kde se tento adresář fyzicky nachází. Uvedený příklad je aktuálně používaný při psaní této knihy a zkoušení příkladů. Následující část `<Location>` je zbytná a používám ji k omezení přístupu k CGI skriptům z bezpečnostních důvodů. Použil jsem dvě metody, omezení přístupu jen na vyjmenované uživatele kteří se prokazují znalostí hesla a omezení přístupu podle (ip) adresy uživatele (jeho počítače).

```
# /etc/init.d/apache-ssl reload
```

Nyní po „reloadu“ konfigurace apache-ssl si můžeme ukázat slibovaný jednoduchý skript.

Příklad 60-3. Jednoduchý CGI skript

```
#!/bin/bash
# $Id: ahoj,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $
echo "Content-type: text/plain"
echo
echo "Ahoj"
```

Tento velmi jednoduchý skript použijeme k tomu abychom si ověřili, že máme správně nakonfigurovaný web server. Jestli ano, měli bychom vidět v prohlížeči po odkazu na náš skript text „Ahoj“. Prvního čeho si všimnete, že náš skript nevrací stránku v html kódu ale v čistém (plain) textu. To není proto, že bychom html nepoužili. V první části našeho skriptu totiž vypisujeme hlavičku a v ní je uvedeno že následuje „text/plain“. Hlavička slouží k informování prohlížeče o tom jaká data budou následovat, a je od těla stránky/dokumentu oddělena prázdným řádkem.

Do prohlížeče zadáme URL `https://localhost/bash/cgi-bin/ahoj` a ukáže se nám výsledek práce našeho jednoduchého CGI skriptu.

* ***FIXME:****vložit případný obrázek s prohlížeče.*

Než si ukážeme nějký html kód, povíme si jak jsou našemu CGI skriptu předány parametry. Tyto webowý server uloží do proměnných v prostředí. Místo dlouhého povídáni příklad.

Příklad 60-4. Proměnné prostředí

```
#!/bin/bash
# $Id: env,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $
echo "Content-type: text/plain"
echo
echo "Prom<65533>nn<65533> prost<65533>ed<65533>:"
env
```

V prohlížeči pak vidím všechny proměnné prostředí. Vetšinu z nich pro nás připravil webový server a můžeme je ve svém CGI skriptu použít.

FIXME: Doplnit parametry předávané v URL.

60.19.13.1.1. Proměnné prostředí

Odkazy:

- 2.2 Using Environment Variables (http://www.oreilly.com/openbook/cgi/ch02_02.html)

Proměnné prostředí jsou jedním ze dvou informačních kanálů kterými komunikuje web server s CGI aplikací. Konkrétně web server nastavuje tyto proměnné ve kterých jsou CGI aplikaci přístupny všechny důležité informace o dotazu od uživatele sedícího u internetového prohlížeče. Rovněž jsou zde uloženy další informace jako jsou informace o počítači uživatele a jejo jméno a způsob autorizace, byl ly nějaký použit www serverem. Tyto

informace pak mohou být doplněny dalšími jenž CGI aplikace přečte ze standardního vstupu (`stdin`) přes který je www server předá.

Proměnné CGI prostředí

`GATEWAY_INTERFACE`

The revision of the Common Gateway Interface that the server uses.

`SERVER_NAME`

The server's hostname or IP address.

`SERVER_SOFTWARE`

The name and version of the server software that is answering the client request.

`SERVER_PROTOCOL`

FIXME:

`SERVER_PORT`

FIXME:

`REQUEST_METHOD`

FIXME:

`PATH_INFO`

Cesta zadaná do prohlížeče.

`PATH_TRANSLATED`

Přeložená cesta mapovaná na lokální filesystém.

`SCRIPT_NAME`

FIXME:

`DOCUMENT_ROOT`

Kořenový adresář www dokumentů. Nejčastěji `/var/www/`

`QUERY_STRING`

FIXME:

`REMOTE_HOST`

FIXME:

`REMOTE_ADDR`

FIXME:

`AUTH_TYPE`

FIXME:

`REMOTE_USER`

FIXME:

`REMOTE_IDENT`

FIXME:

`CONTENT_TYPE`

FIXME:

`CONTENT_LENGTH`

FIXME:

`HTTP_FROM`

FIXME:

`HTTP_ACCEPT`

FIXME:

`HTTP_USER_AGENT`

FIXME:

`HTTP_REFERER`

FIXME:

60.19.13.1.2. Hlavička odpovědi
Response Headers

Jedná se o hlavičky jenž vrací server v odpověd na dotaz od klienta/browseru. Kompletní seznam všech hlaviček je publikovaný na webu w3c jako dokument Object MetaInformation (http://www.w3.org/Protocols/HTTP/Object_Headers.html).

Hlavička je odesílána serverem jako první. Každý řádek hlavičky popisuje jeden parametr. Ukončena je pak prázdným řádkem za kterým může následovat tělo dokumentu.http://www.w3.org/Protocols/HTTP/Object_Headers.html

Platné HTTP hlavičky:

Content-length

Délka odpovědi v bytech.

Content-type

Typ odpovědi. („text/html“, „text/plain“)

Expires

Datum a čas do kdy je dokument platný.

Location

Přesměrování na jiný server.

Pragma

Slouží k ovládání cacheování dokumentů.

Status

Status odpovědi.

Refresh

FIXME: Client reloads specified document.

Set-Cookie

Klient/Prohlížeč uloží specifikované informace. Používá se k sledování prohlížeče/uživatele mezi dotazy.

60.19.13.2. Jednoduchý formulář

Nyní když víme jak funguje předávání parametrů do skritpu, můžeme se pokusit o jednoduchý formulář.

Příklad 60-5. Jednoduchý formulář

```
#!/bin/bash
# $Id: simple-form,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $
echo "Content-type: text/html"
echo


cat <<EOF
<html>
  <head>
    <title>Jednoduch<65533> formul<65533><65533></title>
    <meta http-equiv="Content-Type"
          content="text/html; charset=iso-8859-2"/>
  </head>
  <body>
    <form>
      Zadej jm<65533>no:
      <input type="text" name="jmeno"/>
      <br/>
      <input type="submit" name="tlacitko" value="Odeslat"/>
    </form>
  </body>
</html>
EOF
```

60.19.13.3. Komponenty

Jako komponenty rozumíme znovupoužitelné bloky kódu vykonávající určitou funkci a řešící určitý problém. Z pohledu vývojáře jsou to prefabrikáty/cihly z nichž staví weby, nebo další komponenty. V této části rozeberu několik způsubů od těch jednodušších až k poměrně velmi sofistikovaným.

60.19.13.3.1. Jednoduché komponenty

Nejjednodušší komponenta je proměnná nebo samostatná funkce. Tímto způsobem můžeme vkládat ho jednoho kódu kód jiný. Tuto primitivní metodu můžeme použít s výhodou u takových komponent které nemají vnitřní stav (proměnné) jenž je třeba uchovávat mezi stránkami. Mohou to být třeba komponenty prezentující stav systému, či jinou aktuálně zjišťovanou hodnotu. Mohou prezentovat i data z databáze.

* ***FIXME:****Ukázat příklady takových jednoduchých komponent.*

60.19.13.3.2. Komponenty s vnitřním stavem

Pokročilejším druhem jsou komponenty jenž mají vnitřní stav uložený v proměnných který je potřeba zachovat při přechodu na jinou stránku s toutéž komponentou. Základní otázkou je, jak a kde uchovávat tento stav (hodnoty proměnných). Jednou z možností jsou skrytá pole ve formuláři, či přidané parametry v URL cgi skriptu.

Použijeme-li pro uchovávání skrytá pole

```
<input type="hidden" name="stav_komponenty" value="$stav_komponenty"/>
```

```
<input type="hidden" name="stav_komponenty"
                     value="$stav_komponenty"/>
```

Na toto řešení jsou kladeny následující omezení:

- veškeré zmeny stavu komponenty, t.j. veškeré výpočty, proběhnou před vlatním vykreslováním komponenty a vypočtený stav je uložen ve skrytém poli
- budeme-li mít na stránce více než jednu komponentu, musí tagy `<form>` a `</form>` generovat šablona stránky protože stavové proměnné všech komponent **musí** být uloženy v jednom formuláři. Přesněji všechny formulářové prvky musí být v jednom formuláři. Důvodem je to, že při odeslání dat z formuláře se odešlou jen data toho formuláře, jehož tlačítko bylo stisknuto. Ostatní formuláře se neodesílají a stav v nich uložený by tedy nebyl zachován a přenesen do další stránky.

Dopracujeme se tedy ka komponentám jenž jsou prezentovány dvěma funkcemi.

- `komponenta_state_machine`
- `komponenta_render`

První z těchto funkcí, `..._state_machine`, ukládá stav do skrytých polí a je volána šablonou stránky na začátku pro zapsání skrytých polí do formuláře vytvářeného touto šablonou. V této funkci proběhnou též všechny výpočty jenž z původního stavu a parametrů předaných formulářem či URL spočtou stav nový jenž bude uložen do skrytého pole. Druhá, `..._render`, je volána v okamžiku kdy je třeba vykreslit komponentu.

Mají-li komponenty ovlivňovat hlavičku stránky, musí jejich výpočet proběhnout ještě před vypsáním této hlavičky. Šablona stránky pak musí spusti výpočet stavu komponent a výstup uložit do proměnné aby pak mohla být skrytá pole uložena do tšla stránky. Mají-li se komponenty ovlivňovat navzájem pak nastanou problémy s pořadím v jakém probíhají výpočty. Komponenta která ovlivňuje ostatní musí být spočtena před těmito komponentami.

60.19.13.3.3. Poznámky

FIXME:

```
function register_component() {
    # Ad component name stored in $1 to array
    # of registered components
}
```

```
function compute_components() {
    for component in $components; do
        ${component}_state_machine
    done
}
```

60.19.13.4. Poznámky z projektu NetCFG

Knihovna a cgi programy sdružené do tohoto projektu netcgi vznikly jako experiment jenž započal napsáním jednoduchého skriptu a posléze publikováním dat z konfiguračních databází které používám na svých serverech. Posléze jsem dospěl k názoru že při použití vhodných technologií a určité disciplíně lze napsat v bashi i netriviální aplikaci. O vyvinutí příslušných komponentních/objektových technologií se v tomto projektu pokusím.

Radek Hnilica --

Deník vývojáře a popis projektu

60.19.13.4.1. Jednoduchá šablona stránky

Jednou z prvních věcí kterou jsem udělal bylo napsání šablony stránky. Tato slouží k tomu abych oddělil vnější 'design' od vnitřního obsahu stránky. V první fázi se jedná o triviální způsob kdy šablona, tedy vnější obal, je tvořena dvěma funkcemi. První vypíše do stdout 'hlavičku' tedy část html kódu před tělem stránky a druhá pak na závěr 'patičku' jenž dokončí obalení těla stránky. Princip je následující. Pokud zvolíme například jednoduchý tabulkový design kde stránka má graficky vypadat takto

```
+---------------------------------------+
| logo |      titulek            | user |
|------+-------------------------+------|
| Menu                                  |
|---------------------------------------|
|                                       |
|        obsah                          |
|                                       |
|                                       |
|                                       |
|---------------------------------------|
| rcsinfo                    copyright  |
+---------------------------------------+
```

Pak html kód stránky:

```
<html>
        <head>
          <title>titulek</title>
        </head>
        <body>
          <table>
            <tr>
              <td>logo</td>
              <td>titulek</td>
              <td>user</td>
            </tr>
```

```
      <tr>
        <td>menu</td>
      </tr>
      <tr>
        <td>
------------------------------
            obsah
------------------------------
                </td>
      </tr>
      <tr>
        <td>rcsinfo  copyright</td>
      <tr>
    </table>
  </body>
</html>
```

Tak tento vzor rozdělíme v místě obsahu na části jak napovídají čáry. První část kódu bude vložena do funkce 'template_header' a poslední do funkce 'template_footer' například tímto způsobem

```
function template_header() {
    cat <<EOF
    ...html kód první části...
EOF
}

function template_footer() {
    cat <<EOF
    ...html kód poslední části...
EOF
}
```

Kód generující jednotlivé stránky pak bude vpadat takto

```
function page_prvni() {
    template_header
    ... generování obsahu stránky ...
    template_footer
}
```

Tento primitivní způsob použití šablony stránky rozdělené na hlavičku a patičku nám v začátku postačí a přitom nám dá dostatečně silný nástroj na vytváření webu s potřebnou úpravou. Zásah do hlavičky/patičky se pak projeví na všech vytvářených stranách.

60.19.13.4.2. Podpůrná http knihovna bashlib

Tato knihovna, kterou jsem nalezl na internetu, slouží k analýze/parsování parametrů předávaných z formulářů. Jedním z jejích úkolů je dekódovat tyto parametry, neb řada znaků není předávána přímo ale kódována v hexadecimální notaci %FF. Abychom se tímto nemuseli zatěžovat, použijeme knihovnu bashlib. Knihovna se používá tak, že na začátku CGI skriptu ji načteme pomocí

```
. bashlib
```

nebo

```
source bashlib
```

V tomto okamžiku dojde k analýze parametrů v prostředí a k dekódování těchto. Poté jsou v knihovně definovány funkce k použití v našem CGI skriptu. Jsou to

```
version         vrátí textový řetězec popisující verzi knihovny
version_html    vrátí html kód popisující verzi knihovny
param name      vrátí obsah/hodnotu uvedeného parametru
cookie          -
set_cookie      -
send_redirect   -
```

Poznámka: Nepoužívám verzi z SourceForge (http://bashlib.sourceforge.net/) ale verzi upravenou. Až budu mít chvíli, připravuji podle této verze napsat vlastní knihovnu s názvem `cgilib` nebo `cgi.lib` (bashcgi, bashcgi.lib, ...).

60.19.13.4.3. Použití komponent

Teprve s použitím komponenet začíná být vytváření/programování stránek zajímavé. Právě komponenty ná dovolí sestavovat stránku z připravených prefabrikovaných stavebních kamenů aniž bychom se v tu chvíli dívali na samotnou realizaci komponent. Jedná se tedy o další stupeň abstrakce.

Komponenty píši do samostatných souborů jenž mají charakter knihoven. Každá komponenta tak nabízí ven řadu funkcí přes než se integruje do stránky. Z mého pohledu je konponenta v podstatě samostatný objekt s vlastními datovými poli, vlastním kódem a odpovídající stavovovu proměnnou. Protože bash není objektovým jazykem a jeho možnosti jsou omezené, považují komponenty za objekty typu singleton. Abych oddělil jmenné prostory pro funkce a proměnné, každá komponenta má svůj jedinečný prefix kterým začínají jména všech jejích funkcí a proměnných.

Základní funkcí je funkce zobrazení/renderování komponenty jejímž výstupem je html kód který se vkládá do stránky. Každá komponenta tedy musí nabízet nejméně jednu funkci a to je funkce

```
komponenta_render
```

Nejjednodušší komponenta může tedy vypadat takto:

```
# komponenta
       function komponenta_render() {
           cat <<EOF
           Hello world!
       EOF
       }
```

Takováto komponenta je samozřejmě velmi primitivní a statická, postrádá jakoukoliv dynamicitu v chování. To ovšem neznamená že je zbytečná. Například budeme potřebovat zobrazi stav odpovídající nějaké periferii, obsahu databáze či nejakého příznaku. V takovém případě komponenta jen zapouzdří kód jenž tuto informaci zjišťuje a formuje html kód ji popisující.

60.19.13.4.3.1. Komponenta s vnitřním stavem

Takováto komponenta je pro nás již zajímavější. Má svůj vnitřní stav který se mezi jednotlivými voláními/zobrazováním stránky může měnit a nabízí nám taky základní možnosti interakce s komponentou. Opět komponenta navenek prezentuje renderovací funkci která je již ovšem vnitřně složitější.

```
function komponenta_render() {
          local -r state=$(param komponenta_state)
```

```
        case $state in
                  stav-první)
                      # zobraz komponentu
         ;;
                  stav-druhý)
                      # zobraz komponentu jinak
                      ;;
                  *)
                      # stav hlaví, implicitní
                      # zobraz komponentu
                      ;;
              esac
          }
```

Principem fungování je tedy že v každém stavu zobrazuje komponenta jinak, jiným html kódem. Jistě jste si všimli, že stav komponenty se čte z parametru komponenta_state. Aby vše fungovalo tak ja chceme, musíme zajistit že stav komponenty bude v CGI prostředí, tedy že bude předán do stránky buď to formulářem nebo parametrem v URL. Toho dosáhneme například tak, že komponenta sama vkládá do svého html kódu skryté pole.

```
<form> <!--Formulář zahrnující všechny komponenty-->
      ...
 <input type="hidden" name="komponenta_state"
                  value="stav"/>
      ...
     </form>
```

Protože při stisku tlačítka jsou odeslána jen data v tomto formuláři, musí být všechny komponenty na jednom formuláři. Toho dosáhneme tak, že komponenty samy nevypisují <form> a </form> tagy ale musíme se o vypsání těchto tagů postarat sami.

60.19.13.5. Nezpracované texty

FIXME:

60.19.13.5.1. Pruhovaná tabulka

Pokud nám nevyhovuje standardní čárami rozdělená tabulka, můžem použít design tabulk bez úplně bez čar či jen s veknovním orámováním kdy jednotlivé řádky jsou opticky odděleny odlišnou barvou podkladu. Taková tabulka je často lépe čitelná než čárami rozdělená. Pro dosažení efektu budem potřebovat rozlišovat liché a sudé řádky. Potřebuji dosáhnout tedy následujícího html kódu.

```
<table rules="none" align="center">
  <tr class="even"><td>první řádek</td><tr>
  <tr class="odd"><td>druhý řádek</td><tr>
  .
  .
  .
  <tr class="even"><td>předposlední řádek</td><tr>
  <tr class="odd"><td>poslední řádek</td><tr>
</table>
```

Já používám následující šablonu.

```
row=0
echo "<table rules=\"none\" align=\"center\">"
for i in první druhý třetí čtvrtý pátý; do
    let row++
    if (($row%2)); then
        echo '<tr class="odd">'
    else
        echo '<tr class="even">'
    fi
    echo "<td>$row</td>"
    echo "<td>$i</td>"
    echo '</tr>'
done
echo "</table>"
```

Ačkoliv bych mohl vložit kódy barev přímo do tagů `<tr>`, použil jsem raději třídy a barvy definuji v Style Sheetu takto:

```
# Pruhovaná tabluka
tr.odd  { background-color: #dfd; }
tr.even { background-color: #eee; }
```

60.19.14. Bezpečné programování v bashi

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

Odkazy:

- Security-specific Programming Errors (Part 7) (http://www.linuxkp.org/en/content.php?&content/programming/secprog7.htm
- **FIXME:**
- **FIXME:**
- **FIXME:**

Bezpečné programování je způsob jak neumožnit útočníkovi zneužít náš program/skript k získání neoprávněného přístupu či vykonání nevhodného kódu na našem systému.

```
⋮
CHECK=${1//[0-9a-zA-Z]/}

if [ "$CHECK" = "" ]; then
     echo "It's all fine!"
else
     echo "Namespace is not clean!"
     exit 1
fi
⋮

⋮
trap "echo 'Good Bye';exit 0" 1 2 3 4 5 6 7 8 9 10 11 12 13
                              14 15 16 23 24 25 26 27
⋮
# do something
```

⋮

60.19.15. Nezařazené texty

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

60.19.15.1. Použití příkazu shift

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
until [ -z "$1" ]  # Until all parameters used up...
do
    echo -n "$1 "
    shift
done
echo    # Extra line feed
```

60.19.15.2. Přejmenování přípony souboru

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
mv $filename ${filename%$1}$2
```

60.19.15.3. Čtení ze vstupu ve smyčce

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
while read line
do
    echo "$line"
done <data-file

while IFS=: read name passwd uid gid fullname ignore
do
    echo "$name ($fullname)"
done </etc/passwd
```

60.19.15.4. Přidávání prvků do pole/seznamu/zásobníku

* *rcsinfo="$Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $"*

```
function transient  {
   remove[${#remove[*]}]=$1
}
```

```
transient remove              # remove may not be other than first transient
transient transient
...
# remove transients
i=${#remove[*]}
while [ $i -ne 0 ] ; do
   unset ${remove[--i]}
done
```

Kapitola 61. awk

Odkazy:

- My favorite AWK (http://web.cecs.pdx.edu/~timm/dm/gawk101.html)

AWK vznikl jako nástroj usnadňující zpracování textu na UNIXu.

Myslím, že AWK je v současné době nástroj spíše opomíjený a nedoceňovaný.

`awk` je skriptovacím nástrojem pro spracování textů. Je dalším z řady mocných nástrojů UNIXu. Jedná se o nástroj kombinující v sobě jednoduchým způsobem regulární výrazy a jednoduchý imperativní progamovací jazyk. Ve své podstatě skript/program v jazyce awk sestává z řady pravidel tvaru

vzor { *akce* }

vstupní textový soubor se pak prochází řádek po řádku a porovnává se vzory uvedenými ve skriptu. Pokud vzor vyhovuje, provede se akce.

Pro lepší pochopení jeden příklad. Potřebujeme vytisknout názvy účtů uživatelů s identifikačním číslem uid větším než 1000 včetně. Skript tedy bude vypadat takto:

```
$3>=1000 {print $1}
```

Protože je velmi krátký, přímo jednořádkový, nebudeme jej psát do souboru ale předáme jej příma jako parametr programu `awk`

```
$ awk -vFS=: '$3>=1000 {print $1}' /etc/passwd
nobody
radek
stana
saib
```

Jak je patrné již z tohoto jednoduchého příkladu, **awk** použijeme všude tam, kde s jednoduššími nástroji nevystačíme.

61.1. Zajímavé funkce

Definice funkcí které v AWK nejsou a seznávám je velmi užitečnými.

Příklad 61-1. Funkce pro ořezání zbytečných mezer okolo řetězce `trim`

```
function trim(s,    t) {
        t=s;
        sub(/^[ \t\n]*/, "", t);
 sub(/[ \t\n]*'/, "", t);
        return t
}
```

Kapitola 62. m4

Odkazy:

- GNU M4 (http://www.gnu.org/software/m4/)
- Introduction to m4 (http://www.cs.utah.edu/dept/old/texinfo/m4/m4.html)
- m4 (http://en.wikipedia.org/wiki/M4_(computer_language)) na Wikipedii
-

m4 je makroprocessor.

Kapitola 63. Perl

Odkazy:

- Fundamentals of Perl (http://www.pti.co.il/talks/Fundamentals/index.html)

* *TODO: Odstavec přepracovat a udělat z něj abstrakt kapitoly.*

Vzhledem k vyspělosti Perlu a jeho oblibě a rozšíření po celém světě a velmi velkému výběru literatury, nechci psát další knihu o Perlu. Nepoužívám jej s dostatek abych se vůbec cítil být povolán napsat byť jen malý článek. Tato kapitola je tedy jen sbírkou odkazů na zajímavé materiály a sbírkou ukázek které mne zaujaly nebo jsem je potřeboval k práci.

Řešení:

- Analýza (parsování) .INI souborů.
    - Unix Review Column 40 (Dec 2001) (http://www.stonehenge.com/merlyn/UnixReview/col40.html)

63.1. CPAN

Pokud už nějaká verze CPAN k dispozici je, můžeme jej využít k celé instalaci. Já jsem měl na jednom systému se SuSE 9.0 problémy spustit CPAN. Vyřešil sjem to tak, že požadovaný soubor `Hostname.pm` jsem prostě vzal v aktuální verzi Debian Etch a umístnil do cesty tak aby ho perl našel. Použil jsem adresář `/usr/lib/perl/5.8.8/Sys` který jsem pro tento účel musel vytvořit.

```
# perl -MCPAN -e 'install Bundle::CPAN
```

V průběhu instalace a konfigurace balíčků po vás program může chtít nějaká rozhodnutí. Implicitní volba většinou stačí. Po instalaci, která trvá nějakou dobu, je možno začít s CPAN plnohodnotně pracovat. Můžeme instalovat balíčky

```
# perl -MCPAN -e 'install Bundle:libnet'
...
```

Můžeme také spustit interaktivní shell CPANu a pracovat v něm.

```
# perl -MCPAN -e shell
```

63.2. Práce s daty a časem

Odkazy:

- Dates and Calendars in Perl (http://lexington.pm.org/meetings/dates1.html)
- The Many Dates and Times of Perl (http://www.perl.com/lpt/a/718)

Funkce pracující s datem a časem:

- `localtime`
- `gmtime`

```
($sec,$min,$hour, $mday,$mon,$year, $wday,$yday,$isdst) = localtime(time);

$wday = (localtime(time))[6];
$day_of_week = ('neděle', 'pondělí', 'úterí', 'středa', 'čtvrtek', 'pátek', 'sobota')[$wday];
print "Dnes je $day_of_week.\n";
```

Tabulka 63-1.

		rozsah hodnot
`$sec`	sekundy	0-60
`$min`	minuty	0-59
`$hour`	hodiny	0-23
`$mday`	den v měsíci	1 - 31
`$mon`	měsíc v roce	0-11, 0=leden, 1=únor, . . . , 11=prosinec
`$year`	rok od 1900	
`$wday`	den v týdnu	0-6, 0=neděle, 1=pondělí, . . . , 6=sobota
`$yday`	den v roce	1-366
`$isdst`	Dayilight Saving Time	0 = není letní čas, 1 = letní čas

Příklad 63-1. Jednořádkové ukázky

```
$ perl -e 'print time'
1212660430
```

Příklad 63-2. .

```
#!/usr/bin/perl
@timeData = localtime(time);
print join((' ', @timeData);
```

Příklad 63-3. noname

```
#!/usr/bin/perl
@months = qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);
@weekDays = qw(Sun Mon Tue Wed Thu Fri Sat Sun);
($second, $minute, $hour, $dayOfMonth, $month, $yearOffset, $dayOfWeek, $dayOfYear, $daylightS
$year = 1900 + $yearOffset;
$theTime = "$hour:$minute:$second, $weekDays[$dayOfWeek] $months[$month] $dayOfMonth, $year";
print $theTime;
```

Příklad 63-4. time.pl

```
#!/usr/bin/perl
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=localtime(time);
printf "%4d-%02d-%02d %02d:%02d:%02d\n",$year+1900,$mon+1,$mday,$hour,$min,$sec;
```

63.3. Síťové programování

63.3.1. Tcp server

Odkazy:

- TCP Servers with IO::Socket (http://perl.active-venture.com/pod/perlipc-tcpserver.html)
- Perl, Sockets and TCP/IP Networking. (http://www.perlfect.com/articles/sockets.shtml)
-

Kapitola 64. Zpracování textů

Odkazy:

- An A-Z Index of the Bash command line for Linux (http://ss64.com/bash/)

UNIX vznikl původně pro potřebu patentového oddělení firmy AT&T, pro zpracování textů. Tomu odpovídá velké množství nástrojů pro tento druh práce. Základním nástrojem je shell a jeho schopnost vatvářet kolony příkazů. To umožňuje průběžné zpracování textu kdy každý program v koloně jako stroj na běžícím pásu provede svoji operaci s textem a výsledek pošle ke zpracování následujícímu programu.

* ***$ dpkg -L coreutils***

Krátký přehled některých programů pro zpracování textu

cat
tac

Příkaz **cat** je určen pro čtení textu ze souboru. Je to filtr který neprovádí vůbec nic, tedy to co přečte na vstupu pošle beze změny na výstup. Umožňuje nám ovšem spojovat soubory. Uvedeme-li jako parametr seznam souborů jsou tyto postupně poslány na standardní výstup.

```
$ cat soubor1 soubor2 soubor3 > vysledek
```

Příkaz **tac** funguje velmi podobně s tím rozdílem že soubor čte řádek po řádku od konce. Obrátí tedy pořadí řádků.

```
$ tac
první řádek
druhý řádek
třetí řádek Ctrl+D
třetí řádek
druhý řádek
první řádek
```

wc

Spočte písmena, slova a řádky ve vstupním souboru.

tr

Provádí jednoduchý překlad/transformaci při které umožňuje nahrazovat znaky. Nahrazuje znaky 1 ku jedné. Například následujícíc zápis provede změnu všech malých znaků za velké.

```
$ tr 'a-z' 'A-Z'
```

sed

Proudový editor. Jedná se o obdobu editoru **ed** určenou pro použití v koloně příkazů pro automatickou editaci textu.

head
tail

Tyto dva příkazy pouští na výstup část vstupu podle parametrů. V základním použití pouští **head** uvedený počet řádků od začátku zatímco **tail** od konce. Se sofistikovanějším nastavením můžeme dosáhnout takových efektů jako je výstup všech řádků mimo prvních 8 a podobně.

sort

* *How do I sort a file in place using bash shell? (http://stackoverflow.com/questions/146435/how-do-i-sort-a-file-in-place-using-bash-shell)*

Setřídí řádky na vstupu podle zadaného kritéria. Kritériem je určení podle které části řádku se má třídit. Zdali se má třídit číselně či abecedně a také můžeme třídit obráceně. Program má řadu parametrů ze kterých bych připoměl `-n` jenž třídí čísleně, takže 1 se řadí před 10.

Při třídění je třeba mít v patrnosti jaké je nastavné prostředí. Pokud se nám řadí číslice až za písmena, mám čské prostředí. Chceme-li číslice před písmeny, tedy pořadí (00, 01, 0A) stačí nastavit proměnnou LANG

```
$ LANG=C sort soubor
```

tee

Tee podle písmene 'T' slouží jako odbočka v koloně příkazů. Funguje tak že co je na vstupu pošle na výstup stejně jako příkaz **cat** ale navíc to ještě zapíše do souboru.

```
$ ...|tee meziprodukt|...
```

split

Rozdělení souboru na menší. Či zápis vstupního proudu do více menších souborů. Zamýšlené použití bylo rozdělit velký soubor na menší například pro potřeby transportu na malých médiích.

cut

Remove sections from each line of files. Print selected parts of lines from each file to standard output.

ptx

Produce a permuted index of file contents. Output a permuted index, including context, of the words in the input files.

uniq

nl

Očísluje všechny řádky. Tedy na výstup pošle co je na vstupu a na začátek každého řádku přidá jeho pořadové číslo. Program má hodně voleb které dovolují například číslovat jen část řádků a mnohé další.

fold

Zalamování dlouhých řádků v souboru. Toto je jednoduchý nástroj, který rozdělí všechny dlouhé řádky podle zadaných kriterií. Jsou to maximální délka (`-w 24`) rozdělovat jen v mezeře mezi slovy (`-s`).

Varování

Nesrovná se s češtinou v UTF-8 kódování.

comm
diff

Porovnání dvou souborů.

cksum
sha1sum
sha224sum
sha256sum
sha384sum
sha512sum

join

paste

* *Lesser-known Linux commands: join, paste, and sort (http://articles.techrepublic.com.com/5100-10878_11-5031653.html)*

csplit

expand
unexpand

fmt

Simple optimel text formater. Reformat each paragraph in files.

pr

Convert text files for printing. Paginate or columnate files for printing.

pv

* *http://www.ibm.com/developerworks/aix/library/au-spunix_pipeviewer/?S_TACT=105AGY20&S_CMP=HP*

Pipe Viewer je program který měří informace protékající rourou a umožnuje zobrazovat jejich aktuální hodnoty. Dovede například vytvářet teploměry ukazující stav rozpracovanosti.

bfr

* *bfr(1) - Linux man page (http://linux.die.net/man/1/bfr)*

unbuffer

* *Turn off buffering in pipe (http://stackoverflow.com/questions/1000674/turn-off-buffering-in-pipe)*

Nejsilnější stránkou je možnost kombinovat příkazy do kolon. Můžeme například očíslovat řádky ve zdrojovém souboru, zalomit aby se vešly na stránku (na šířku stránky či sloupce) a hotové umístnit třeba ve dvou sloupcích na papír s tím že máme ještě očíslované stránky.

```
$ cat source.c| nl | fold -w64 | pr -2 -w136 >k-tisku
```

Velmi jednoduché dvousloupcové sazby textu dosáhneme například takto:

```
$ cat text| fmt -w 24 | pr -2 -w 52
```

Z mých pokusů jsem si všiml že český text kódovaný v utf-8 se neformátuje správně. V první chvíli jsem to obešel překódováním do jiného kódování které nepoužívá vícebytové kódování.

```
$ LANG=cs_CZ recode utf8..l2 <text|fmt -w25|pr -2 -w54|recode l2..utf8
```

Ale pro správný provoz to bude vyžadovat ještě nějaké vyladění.

Kapitola 65. make

* ***Attributy:*** *id="make" xreflabel="make"*

Odkazy:

- **make** Manual ()
- **FIXME:** ()

Program make je zajímavý nástroj pro řešení řady problémů jenž se vyskytují zejména při programování a zpracován dat. Jeho základním principem je řešení závislostí. Jeho běh je řízen souborem pravidel. Tento soubor se jmenuje `GNUmakefile`, `Makefile`, `makefile`, případně si můžeme s pomocí parametru na příkazovém řádku vynutit soubor jiného jména.

```
cíl:    závislosti ...
      příkazy
      ...
```

65.1. Vestavěné proměnné

FIXME:TBD

Tabulka 65-1. Rychlý přehled proměnných v Makefile

proměnné	popis obsahu
`$@`	cíl pravidla který se právě uskutečňuje
`$%`	
`$<`	první ze seznamu závislých souborů
`$?`	závislé soubory které jsou novější než cíl
`$^`	všechny závislé soubory
`$+`	všechny závislé soubry v četně duplikací
`$*`	obsahuje cíl bez přípony

`$@`

Obsahuje cíl pravidla. Tedy ten ze souboru cílů který je právě uskutečňován, je-li v pravidle cílů více.V následující ukázce tedy obsahuje `bar.c`.

```
foo.o: bar.c
 $(CC) -c $< -o $@
```

`$%`

* *PŘELOŽIT: The filename element of an archive member specification.*

`$<`

Obsahuje první ze seznamu závislých (prerequisite) souborů. V následující ukázce tedy obsahuje `rub.c`.

```
a.o: rub.c sat.c bond.h
 echo $<
```

`$?`

Seznam závislých (*prerequisite*) souborů, které jsou novější než cíl. Jednotlivá jména souborů jsou oddělena mezerou.

```
a.o: rub.c sat.c bond.h
 @echo $?
 @echo "\$$? = $?"
```

```
$ touch a.c sat.c
$ make a.o
sat.c bond.h
$? = sat.c bond.h
```

`$^`

Seznam **všech** závislých souborů. Jednotlivá jména jsou oddělena mezerou.

```
a.o: rub.c sat.c bond.h
 @echo "\$$^ = $^"
```

```
$ touch a.c sat.c
$ make a.o
$^ = rub.c sat.c bond.h
```

`$+`

Obdoba proměnné `$^`. Seznam všech závislých souborů, v četně duplikátů. V normálních situacích proměnné obsahují každý soubor jen jednou, i když jej přiřadíme vícekrát. V případě proměnné `$^` jsou zdvojená jména souborů ponechána.

`$*`

Obsahuje cíl bez přípony. V dříve uvedených ukázkách by to tedy byl soubor `a`.

$(wildcard *`pattern` ...*)

FIXME:dopsat

65.2. Obecná pravidla

*

```
# Generic rules
%.lst: %.pal
 pal $<
%.rim: %.pal
 pal -r $<
%.bin: %.pal
 pal $<
```

65.3. Funkce

Následující ukázka zjistí všechny .pal soubory a vytvoří k nim odpovídající seznamy .rim, .lst a .bin souborů.

```
sources = $(shell ls *.pal)
rims = $(subst .pal,.rim,$(sources))
```

```
lists   = $(subst .pal,.lst,$(sources))
bins    = $(subst .pal,.bin,$(sources))
```

* *Podle knihy Managing Projects with GNU Make, 3rd Edition (http://my.safaribooksonline.com/0596006101/make3-CHP-2-SECT-4) by mělo fungovat i "source = *.pal", ale mě to nefunguje.*

65.4. Poznámky

FIXME: Obsah sekce

Nastavení přípon. První příkaz maže všechny přípony, další nastaví jen ty potřebné.

```
.SUFFIXES:
    .SUFFIXES: .c .o
```

Můžeme použít proměnnou `VPATH`

```
foo.1: foo.man sedscript
    sed -e $(srcdir)/sedscript $(srcdir)/foo.man >$@
```

Standardní cíle které bychom v našem `makefile` měli implemetovat:

- `all`
- `install`
- `uninstall`
- `clean`
- `distclean`
- `dist`
- `check`

Kapitola 66. Autotools

* ***Attributy:*** *id="autotools"*

Balíčky v Debian GNU/Linux

- `autoconf`
- `autogen`
- `automake`
- `autoproject`
- `libtool`

66.1. autoconf

Odkazy:

- Autoconf - GNU Project - Free Software Foundation (FSF) (http://www.gnu.org/software/autoconf/autoconf.html)
-

66.2. automake

Odkazy:

- Automake - GNU Project - Free Software Foundation (FSF) (http://www.gnu.org/software/automake/automake.html)
-

66.3. libtool

Odkazy:

- GNU Libtool - The GNU Portable Library Tool (http://www.gnu.org/software/libtool/)
- GNU Libtool 2.2.6 (http://www.gnu.org/software/libtool/manual/libtool.html)

Kapitola 67. Správa zdrojového kódu (SCM)

Odkazy:

- Karl Fogel's CVS book (http://cvsbook.red-bean.com/)
- CVS User's Guide (http://www.loria.fr/~molli/cvs/doc/cvs_toc.html)
- Another CVS tutorial (http://cellworks.washington.edu/pub/docs/cvs/tutorial/cvs_tutorial_1.html)
- Yet another CVS tutorial (a little old, but nice) (http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/cvs/)
- An old but very useful FAQ about CVS (http://www.cs.utah.edu/dept/old/texinfo/cvs/FAQ/txt)

Jakmile naše programátorské pokusy překročí velmi jednoduché školní příklady, a začněme psát něco pro praktické použití, ocitneme se před otázkou jak udržovat zdrojový kód. Pod údržbou zdrojového kódu rozumím archivaci a rozlišení jednotlivých verzí, jakož to i minimální dokumentaci o těchto verzích. V praxi se nám stane že dostaneme informaci o chybě která se projevuje v starší verzi programu a my stojíme před otázkou kde a co opravit.

Nejprimitivnější způsob správy zdrojového porgramu je prostě daný zdrojový kód celý zazálohovat. Můžeme tak činit podle čísla verze, jestli je používáme a udržujeme, nebo podle data.

```
$ cd projekt/..
$ tar cf - projekt |bzip2 --best >projekt.2007-09-16.tar.bz
$ # nebo
$ tar cf - projekt |bzip2 --best >projekt.číslo-verze.tar.bz
```

Vytvořený zkomprimovaný balíček si pak uložíme třeba do archivu.

```
$ mv projekt.číslo-verze-nebo-datum.tar.bz2 $HOME/archiv/
```

Tip: Pokud používáme program make, můžeme v `makefile` definovat cíl `archivuj`.

```
archivuj:
        cd ..; tar cf - $(PROJEKT) | bzip2 --best >$(ARCHIV)/$(PROJEKT).‘date +%F‘.tar.bz
```

Proměnná `$(PROJEKT)` obsahuje název projektu a proměnná `$(ARCHIV)` pak adresář do kterého se zapisují archivní verze.

Takovýto způsob správy zdrojových kódů je ovšem značně primitivní, má velké nároky na diskový prostor a velmi špatně se v něm orientuje. Proto již v ranných dobách vznikl první systém správy verzí, SCCS. Takovýto systém sleduje samotné změny ve zdrojovém kódu a umožňuje nám říci jak se tento měnil v čase, rozumějte mezi jednotlivými verzemi.

Poznámka: Systém správy verzí je program, či spíše skupina programů, která umožňuje zaznamenat a sledovat celou historii změn ve sdrojových souborech.

Poznámka: UNIX nebyl jediným operačním systémem a SCCS zcela jistě nebyl prvním programem svého druhu. Ale programy jenž nejsou běžně dostupné na Linuxu/UNIXu nejsou předmětem tohoto dokumentu.

Poznámka: Systém správy verzí je program, či spíše skupina programů, které umožňují zaznamenat celou historii zdrojových souborů. Tedy zaznamenat všechny změny kterými soubory prošly s možností získat kter-

oukoliv revizi těchto souborů. Můžeme se tak podívat, jaký byl obsah zdrojových souborů před měsícem, nebo před nějakou důležitou změnou, jakožto si i vyjet rozdíly mezi jednotlivými revizemi.

67.1. SCCS - Source Code Control System

* *section id="sccs" xreflabel="SCCS" status="draft" condition="author"*

SCCS je první program pro zprávu verí o jehož existenci na UNIXu mám informace. Ty pochází z knihy „Operační systém Unix a jazyk C“ od pánů RNDr. Jana Brodského, CSc. a RNDr. Luď ka Skočovského.

Poznámka: Program SCCS je proprietární. Existuje jeho svobodná implementace GNU CSSC. (http://cssc.sourceforge.net).

67.2. RCS - Revision Control System

* *section id="rcs"*

Protože SCCS byl *proprietární*, byl komunitou

* *FIXME: doplnit autora*

napsán tento program. Vychází z podobných principů jako SCCS ale je kompletně jiný.

67.2.1. Klíčová slova a jejich substituce

Klíčová slova jsou slova ohraničená z obou stran znakem ’$’. Při /*FIXME: checkout/ jsou nahrazena příslušným textem který je u každého jiný. Tímto snadno vložíme do zrojového textu informace o revizi, souboru, stavu a řadu dalších.

Klíčová slova ve zdrojovém textu (RCS)

$Author$

Bude nahrazeno přihlašovacím jménem autora. Například `$Author: radek $`

$Date$

Bude nahrazeno datem posledního odeslání změn. Npříklad $Date: 2009-03-07 03:52:40 $

$Header$

Bude nahrazeno některými standardními informacemi záhlaví. $Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $

Id

Bude nahrazeno některými standardními informacemi záhlaví, ovšem bez úplné cesty k souboru. $Id: unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $

$Locker$

$Locker: $

Log

Bude nahrazeno neustále se rozšiřující/zvětšujícím seznamem/deníkem zpráv.

```
$Log: ch-sprava_verzi.xml,v $
Revision 1.14  2003/11/26 12:23:16  radek
CVS vyděleno do samostatného souboru.

Revision 1.13  2003/08/19 08:01:15  radek
Přidán jen odkaz na arch systém

Revision 1.12  2003/05/28 07:38:17  radek
Zapomenutý commit.

Revision 1.11  2002/10/22 19:00:38  radek
Nepravidelný commit

Revision 1.10  2002/10/14 19:35:57  radek
Nepravidelný commit.

Revision 1.9  2002/06/17 14:38:08  radek
Nepravidelný commit

Revision 1.8  2002/06/05 15:51:12  radek
Nepravidelný update

Revision 1.7  2002/04/22 12:33:02  radek
Nepravidelný commit: 2002-04-22 odpoledne
přidán popis použití apt-proxy

Revision 1.6  2002/03/18 16:49:39  radek
Nepravidelný commit: 2002-03-18 večer

Modified Files:
 Makefile ch-disky.xml ch-editors.xml ch-ldap.xml
 ch-linux_start.xml ch-smtp.xml ch-sprava-uctu.xml
 ch-sprava_verzi.xml ch-tcpip.xml ch-xml_sgml.xml
 docbook-fo.xsl print.dsl

Revision 1.5  2002/02/22 14:08:07  radek
Nepravidelny commit 2002-02-22 odpoledne
Modified Files: ch-sprava_verzi.xml

Revision 1.4  2002/02/20 22:12:23  radek
Pravidelný commit.  2002-02-20 večer
Modified Files:
 ch-disky.xml ch-ethernet.xml ch-ldap.xml ch-python.xml
 ch-smtp.xml ch-sprava_verzi.xml ch-tcpip.xml unix.xml
Added Files:
 ch-linux_start.xml
```

$Name$

$Name: $

$RCSfile$

Bude nahrazeno jménem souboru. $RCSfile: unix.xml,v $

$Revision$

Číslo, nebo spíše číselný kód revize souboru. Například $Revision: 1.7 $

$Source$

$Source: /home/radek/cvs/unix-book/unix.xml,v $

$State$

$State: Exp $

67.3. CVS - Concurent Version System

* *section id="cvs" xreflabel="CVS" condition="author"*

Zdroje a odkazy:

- CVS klient Eclipse (Eclipse as a CVS client)
- **FIXME:**

ToDo list

1. **FIXME:**

FIXME:

CVS bylo vytvořeno pro správu verzí v prostředí klient-server. Tedy vlastní databáze/sklad verzí může běžet na jiném počítači v síti. Ve svých počátcích to vlastně byla jen skupina skriptů kolem RCS. Z tohoto pohledu se CVS jeví jako rozšíření RCS do sítě.

Co CVS není, co neumí

- CVS může dělat spoustu věcí, ale ne všechno.
- CVS neumí sestavovat programy, CVS není náhrada za make
- CVS nenahradí management projektu.
- CVS nenahradí komunikace mezi programátory.

Zajímavé manuály jsou k nalezení na www.penguin.cz

- Open Source Development with CVS (http://www.penguin.cz/doc/cvsbook/cvsbook_toc.html)
- The CVS manual (http://www.penguin.cz/doc/cvs/html-info/cvs_toc.html)
- CVS client-server protocol description (http://www.penguin.cz/doc/cvs/html-cvsclient/cvsclient_toc.html)

Další článek je na www.keczy.cz

- Správa projektů pomocí CVS (http://www.keczy.cz/index.php?tisk=104) od Ing Miloslava Ponkráce

```
$ export CVS_RSH=ssh
```

```
$ CVSROOT=$HOME/cvs
$ export $CVSROOT
$ echo $CVSROOT
```

```
$ CVSROOT=:pserver:radek@cvs.foo.org:/var/lib/cvs
$ export $CVSROOT
$ echo $CVSROOT
```

67.3.1. Instalace CVS

Nainstalování provedeme velmi jednoduše

```
# apt-get install cvs
```

Před instalací jsme dotázáni v rámci konfigurace na několik informací

* ***FIXME:*** *doplnit*

Potřebujeme-li CVS rekonfigurovat, spustíme konfiguraci takto

```
# dpkg-reconfigure cvs
```

* ***FIXME:*** *následující text přepsat a opravit.*

Budem-li provozovat CVS „síťově“ zvolíme to při konfiguraci či rekonfiguraci. Systém nám pro tento účel vytvoří skupinu `src`. Uživatel jimž pak chceme cvs server zpřístupnit do této skupiny přidáme.

```
# adduser uživatel src
```

Příklad popužití takovéto síťové repository

```
$ cvs -d server:/var/cvs příkaz
```

67.3.1.1. Vzdálený přístup

Potřebujeme-li přistupovat na cvs server ze vzdáleného počítač můžeme s úspěchem použít přístup přes ssh.

FIXME: popsat

Při obracení se na tokovýto vzdálený cvs server je vhodné před použitím **cvs** nastavit některé proměnné nemusíme potom uvádět přepínače přikazu **cvs** jenž blíže popisují tento vzdálený stroj a způsob přistupu k němu.

```
$ export CVSROOT=cvs.firma.cz:/var/lib/cvs
$ export CVS_RSH=ssh
```

Potřebujeme-li anonymní přístup, musíme použít pserver. Tento nakonfigurujeme při instalaci nebo později spustíme program

```
# dpkg-reconfigure cvs
```

FIXME: Popsat nastavení

Pro větší bezpečnost jsem se rozhodl nespouštět pserver pod uživatelem `root`, ale vytvořil jsem jen pro tento účel uživatele `src`

```
# adduser --system src
```

Poté jsem opravil název uživatele z `root` na `src` v souboru `/etc/inetd.conf`

```
cvspserver stream tcp nowait.400 cvs /usr/sbin/tcpd /usr/sbin/cvs-pserver
```

a převedl vlastnictví celé „repository“ na tohoto nově založeného uživatele `cvs`

```
# chown --recursive src /var/lib/cvs
```

67.3.1.2. Úpravy v instalaci pro vzdálený přístup

Podle CVS na dálku (http://abclinuxu.cz/clanky/ViewRelation?relationId=6089)

67.3.1.2.1. Úpravy v konfiguraci systému

Jako správce serveru modifikujeme soubor `/etc/services` a vložíme do něj řádku

```
cvspserver 2401/tcp
```

pokud používáme démon inetd přidáme do souboru `/etc/inetd.conf` tuto řádku

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

používáme-li démon xinetd vytvoříme soubor `/etc/xinetd.d/cvspserver`

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

V obou případech parametr *--allow-root* ukazuje na CVSROOT který bude zpřístupněn/exportován přes pserver.

> **Poznámka:** Parametr *--allow-root* může být zadán vícekrát a tímto zpřístupněno více repositářů.

Nakonec restartujeme démon inetd či xinetd. Například takto

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

67.3.1.3. Příprava repositáře CVSROOT

Přihlásíme se jako uživatel vlastnící repositář a tento inicializujeme

```
# cvs -d /home/cvs init
```

Vytvoří se adresář `/home/cvs/CVSROOT` se spoustou „administrativních“ souborů. Nevytvoří se ovšem soubor `passwd`. Tento sy vytvoříme sami. Jeho struktura je obdobná souboru `/etc/passwd`. První políčko je jméno, druhé obsahuje zašifrované heslo jenž získame programem crypt a třetí obsahuje skutečného uživatele systému jehož jménem bude cvs provádět změny.

konkrétní obsah souboru `CVSROOT/passwd` může vypadat takto

```
pub::pubcvs
literakl:HTRphPBvKJtjA:literakl
oazanon:TXiF1923PHrtI:oazanon
```

první řádek určuje, že uživatel pub nepotřebuje žádné heslo a je namapován (tedy vystupuje jako) systémový uživatel pubcvs.

Příklad 67-1. Jednoduchý skript pro zašifrování hesel

```
#!/usr/bin/perl
# $Id: cvspassword.pl,v 1.1.1.1 2009-01-24 15:42:51 radek Exp $

srand (time());
my $randletter = "(int (rand (26)) + (int (rand (1) + .5) % 2 ? 65 : 97))";
my $salt = sprintf ("%c%c", eval $randletter, eval $randletter);
my $plaintext = shift;
my $crypttext = crypt ($plaintext, $salt);
print "${crypttext}\n";
```

67.3.2. Práce s CVS

Poznámka: Dopsat základy práce s CVS, najít vhodnější název sekce.

Téměř všechny úkony s CVS se provádejí přes program cvs. Prvním parametrem tohoto programu je CVS příkaz.

Seznam CVS příkazů

- add
- admin
- checkout
- commit
- diff
- edit
- editors
- update
- watch
- watchers

67.3.3. Zakládáme nový projekt

Postupy používané při zakládání nových projektů do CVS a ukázky.

Vložení nového projektu do CVS *repository* se říká importování (*importing*). Provede se takto:

```
$ cd můj-projekt
```

```
$ cvs import -m "log msg" jméno_projektu vendortag releasetag
```

Například zavedení této knihy do CVS vypadalo takto

```
$ export CVSROOT=$HOME/cvs
$ cd document/book/unix
$ cvs import -m "initial import into CVS" unix-book radek start
```

Obrázek 67-1. Nový projekt

```
$ export CVSROOT=$HOME/cvs
$ cd project
$ cvs import -m "initial import ..." project radek start
$ cd ..
$ mv project project.remove
$ cvs co project
```

Poznámka: Soubory v novém projektu jsou založeny s číslem revize 1.1.1.1. Není třeba se toho bát, při první změně se číslo revize změní na 1.2.

67.3.4. Získání (checking out) pracovní kopie

```
$ cvs checkout myproj
```

67.3.5. add - přidání souboru do projektu

```
$ cvs add soubor
```

67.3.6. update - aktualizace pracovní kopie

CVS příkaz **update** slouží k aktualizaci pracovního adresáře/adresářů, jenž jsme získali pomocí příkazu **checkout**. Příkaz nepotřebuje znalost cvs serveru, neb všechny informace důležité pro kontaktování CVS serveru jsou uloženy v `CVS` podadresářích.

```
$ cvs update
```

67.3.7. status

FIXME: dopsat

67.3.8. log

FIXME: dopsat

67.3.9. Klienti CVS

V předchzích částech jsem popisoval prácí s CVS pomocí řádkového programu/klienta `cvs`. Nyní uvedu pár programů s grafickým rozhraním.

67.3.9.1. TkCVS

Velice pěkný klient programovany v Tcl/Tk.

67.3.10. Administrace

Konfigurace CVS je uložena v cvs repository jako jeden s projektů s názvem `CVSROOT`. Administrace tedy probíhá tak že si „vytáhneme“ tento projekt k sobě do pracovní oblasti, provedemé úpravy a posléze příkazem **commit** tuto novou konfiguraci zapíšeme.

```
$ cd ~/src
$ export CVSROOT=cvs.firma.cz:/var/lib/cvs
$ export CVS_RSH=ssh
$ cvs checkout CVSROOT
```

67.3.10.1. Anonymní přístup

Potřebuji-li zajistit anonymní přístup do repository, samozřejmě jen pro čtení, provedu to přidáním účtu se jménem `anonymous` či jiným dohodnutého jménem, které svým uživatelům oznámím, do souboru `CVSROOT/readers`

```
$ echo "anonymous" >>CVSROOT/readers
```

Protože tento soubor původně neexistoval, musím je přidat k souborům projektu

```
$ cvs add readers
```

Takto upravenou konfiguraci zapíšeme zpět do CVS

```
$ cvs commit -m "povolen anonymní přístup uživateli anonymous bez hesla"
```

Na serveru ještě musím vytvořit soubor `CVSROOT/passwd`. Tento nemohu vložit přes **commit** ale musím ho vytvořit na serveru přímo.

```
# echo "anonymous:qaL22Gqu.rhmw" >>CVSROOT/passwd
```

K anonymnímu CVS pak přistupujeme přes pserverver

```
$ cvs -d :pserver:anonymous@cvs.firma.cz:/var/lib/cvs login
```

67.4. Subversion

Zdroje a odkazy:

- Subversion (http://subversion.tigris.org/)
- Best Practices for Subversion (SVN) (http://www.youtube.com/watch?v=MfLLDg7zWQg)

Časté úlohy

Vytvoření osobní repository

```
$ svnadmin --fs-type=fsfs create $HOME/svnroot
```

Vytvoření nového projektu *project*

```
    $ cd /tmp
$ mkdir -p -m 0700 project/trunk project/tags project/branches
$ cd /scripts/trunk
$ vi soubor
$ cd ..
$ svn import /tmp/project file://$HOME/svnroot/project
```

Import na vzdálený server přes ssh tuneling

```
$  svn import /tmp/project svn+ssh://svn@server/home/svn/svnroot/project
```

Získání aktuální verze projektu *project*

```
$ svn checkout file://$HOME/svnroot/project


$ svn info
```

Přidání souboru

```
$ svn add file
```

Uložení změn (commit)

```
$ svn ci
```

Aktualizace zdrojových kódů z repository

```
$ svn up
```

Expanze klíčového slova Id pro soubor `hello.c`

```
$ svn propset svn_keywords "Id" hello.c
```

67.4.1. Instalace

Pod Debianem je instalace velmi jednoduchá.

```
# aptitude install subversion
```

67.4.2. Klíčová slova

Tak jako 67.2 a CVS a ostatní systémy správy verzí, má i Subversion klíčová slova. Tato zapsána ve zdrojovém souboru jsou v okamžiku získávání zdrojového souboru s repository nahrazena hodnotami. Na rozdíl od zmíněných 67.2 a CVS jsou však klíčová slova v Subversion trochu jiná a nedochází k jejich expanzi automaticky. Soubor obsahující klíčová slova musí mít nastaven atribut `svn:keywords`. Hodnotou tohoto adtributu je seznam klíčových slov. Slova zde uvedená se pak expandují a jiná ne.

Nastavení tohoto atributu provedem například takto:

```
$ svn propset svn:keywords "Id,HeadURL" readme
```

O aktuální hodnotě tohoto parametru se přesvědčíme příkazem:

```
$ svn propget svn:keywords readme
```

Date

This keyword describes the last time the file was known to have been changed in the repository, and looks something like $Date: 2002-07-22 21:42:37 -0700 (Mon, 22 Jul 2002) $. It may also be specified as LastChangedDate.

Revision

This keyword describes the last known revision in which this file changed in the repository, and looks something like $Revision: 144 $. It may also be specified as LastChangedRevision or Rev.

Author

This keyword describes the last known user to change this file in the repository, and looks something like $Author: harry $. It may also be specified as LastChangedBy.

HeadURL

This keyword describes the full URL to the latest version of the file in the repository, and looks something like $HeadURL: http://svn.collab.net/repos/trunk/README $. It may be abbreviated as URL.

Id

This keyword is a compressed combination of the other keywords. Its substitution looks something like $Id: calc.c 148 2002-07-28 21:30:43Z sally $, and is interpreted to mean that the file calc.c was last changed in revision 148 on the evening of July 28, 2002 by the user sally.

67.4.3. Postupy

Zde uvedené postupy vychází z prvního pokusu použít svn. Volený způsob práce nemusí být standardní ale vychází z momentálnívh potřeb začít používat svn pro jeden projekt.

Nejdříve si vytvoříme adresář pro všechny repository. (jedna repository = jeden projekt). Je důležité si uvedomit že jeden vytvořený adresář pomocí **svnadmin create** je uzavřený celek. Tudíž jsem dospěl k názoru že pokud budu chtít pracovat s svn podobným způsobem jako pracuji s cvs, musím pro každý projekt vytvořit samostatnou repository. Tak jako mám pro cvs jednu repository v adresáři `$HOME/cvs`, tak jsem pro subversion zvolil kořen všech repository v adresáři `$HOME/svn`. Zde má každý projekt vlastní adresář s vlastní repository.

```
$ mkdir $HOME/svn
```

Vytvoření nové „repository“ pro projekt `projekt`.

```
$ svnadmin create $HOME/svn/projekt
```

Prvotní import projektu

```
$ svn import /path/to/projekt file://$HOME/svn/projekt -m "initial import"
```

Tím máme uloženu prvotní revizi 1 v repository, a můžeme započít práci. Nejdříve si z repository „checkneme“ aktuální verzi:

```
$ svn co file://$HOME/svn/projekt
```

67.4.3.1. Import / Vytvoření projektu

- A Quick Start: Chapter 1. Introduction (http://svnbook.red-bean.com/en/1.1/ch01s07.html)

Nejdříve si musíme připravit dvě věci. Repository a adresář s projektem, začnu repository. Předpokládejme, že všechny svn repository máme jako podadresáře v adresáři `/var/lib/svn/`. Novou repository projektu `snimky` vytvoříme tedy takto:

```
$ svnadmin create /var/lib/svn/snimky
```

Teď si připravíme adresářovou strukturu projektu. Vytvoříme si ji třeba v `/tmp/`. Ta vypadá následovně:

```
/tmp/project/branches/
/tmp/project/tags/
/tmp/project/trunk/
```

```
$ mkdir -p /tmp/snimky/branches /tmp/snimky/tags /tmp/snimky/trunk
```

Do adresáře `/tmp/snimky/trunk` nakopírujeme celý strom zdrojových souborů našeho projektu. Takto vytvořenou adresářovou strukturu importujeme do vytvořené repository.

```
$ svn import /tmp/snimky file:///var/lib/svn/snimky -m "initial import"
```

Po dokončení importu můžeme vytvořené adresáře odstranit. Pokud tak neučiníme, systém je za určitý čas odstraní sám, protože jsme je vytvořili v pracovním adresáři `/tmp/`.

```
$ rm -r /tmp/snimky
```

Celou akci vytvoření projektu dokončíme získáním pracovní verze zdrojových kódů.

```
$ cd ~/src
$ svn co file:///var/lib/svn/snimky
```

FIXME:odstranit/přepracovat zbytek textu sekce.

Nejdříve vytvoříme adresář s projektem.

V adresáři `project/trunk` budou všechny zdrojové soubory projektu. Takto vytvořený projekt importujeme.

```
$ svn import .../project  file://cesta/k/repository -m "initial import"
```

Před použitím projektu, tedy před úpravami zdrojových kódů je třeba získat ze Subversion repository pracovní verzi.

```
$ svn checkout file:///cesta/k/repository/trunk project
```

FIXME:

67.4.4. Poznámky

Pokud chceme používat svn z Emacsu, existují dvě řešení:

> Are there an Emacs hooks for checking in/out svn repos?

Yes, two in fact. See vc-svn.el and psvn/psvn.el in
tools/client-side.

Matt

67.5. SVK

* ***FIXME:****napsat.*

Odkazy:

- Distributed Version Control with svk (http://www.perl.com/pub/a/2004/03/03/svk.html) by Cjia-liang Kao (http://www.perl.com/pub/au/Kao_Chia_liang) on perl.com

67.6. arch

Odkazy a zdroje:

- arch (http://www.regexps.com/#arch)
- Arch Revision Control System (http://arch.fifthvision.net/bin/view)

67.7. Git

Odkazy:

- Git — Fast Version Control System (http://git.or.cz/) (project homepage)
- Git-Wiki (../ruby/git-wiki.html): wiki nad gitem a Ruby
- Ruby and Git Roundup: Rails, Rubyforge, APIs (http://www.infoq.com/news/2008/04/ruby-git-roundup-rails-rubyforge)
- GitHub - Rails-based Git repository hosting (http://www.infoq.com/news/2008/03/github-git-repository-hosting)
- Git User's Manual (http://www.kernel.org/pub/software/scm/git/docs/user-manual.html)
- Everyday GIT With 20 Commands Or So (http://www.kernel.org/pub/software/scm/git/docs/everyday.html)
- Kernel Hackers' Guide to git (http://linux.yyz.us/git-howto.html)
- Git (http://en.wikipedia.org/wiki/Git_(software)#cite_note-44) na Wikipedii
- Git - a Talk by Randal Schwartz (http://video.google.com/videoplay?docid=-3999952944619245780#) na Google videos
- 10 Git Tips and Tricks for Beginners (http://www.developer.com/features/article.php/3886146/10-Git-Tips-and-Tricks-for-Beginners.htm)

Nejdřívě několik rychlovek.

Informaci o konfiguraci gitu a daného repositáře zjistíme když v adresáři repositáře napíšeme příkaz

```
$ git config -l
```

67.7.1. Instalace z balíčků

```
# aptitude install git-core
```

67.7.2. Instalace z Backports.ORG

Do `/etc/apt/sources.list` přidáme zdroj odkazující na Bakports.ORG

```
# echo "deb http://www.backports.org/debian etch-backports main contrib" >>/etc/apt/sources.li
# aptitude update

# aptitude -t etch-backports install git-core
```

67.7.3. Instalace ze zdrojů

Při překladu budeme potřebovad další věci. Mě chyběly balíčky `libcurl3-dev` a `autoconf`.

```
# aptitude install libcurl3-dev autoconf
```

Budeme-li generovat dokumentaci, budem potřebovat další programy. Mě chyběli balíčky `asciidoc` a `xmlto`.

```
# aptitude install asciidoc xmlto
```

Nyní si můžeme stáhnout zdroje, nakonfigurovat, přeložit a vygenerovat dokumentaci. Nakonec provedeme instalaci. Ke stažení gitu jsem použil jeho starší verzi nainstalovanou v systému. Při konfigurování jsem nezadal žádné další parametry, neboť jsem při kontrole konfigurace nezjistil žádné zvláštnosti. Git se nainstaluje do `/usr/local`.

```
$ git clone git://git.kernel.org/pub/scm/git/git.git
$ cd git
$ make configure
$ ./configure
$ make all doc
```

Překlad trval nějakou chvílí, ale vytváření dokumentace bylo podstatně delší. Poté jsem pod rootem dokončil vše instalací programů a dokumentace.

```
# make install install-doc
```

Cesty mám správně, v proměnné `PATH` adresář `/usr/local/bin` předchází adresáři `/usr/bin`, a tak mi vše ihned fungovalo.

```
$ git version
git version 1.5.5.g7134
```

67.7.4. Základní operace s repositářem

67.7.4.1. Vytvoření projektu

Vytvoření prázdného repositáře se provádí příkazem **git init**.

mkdir git *projekt*
cd git *projekt*
git init

Chceme například vytvořit repositář pro projekt, který se jmenuje „stravenky“. Použijeme následující posloupnost příkazů.

```
$ mkdir -p ~/git/stravenky
$ cd ~/git/stravenky
$ git init
Initialized empty Git repository in .git/
```

```
$ mkdir projekt
$ cd project
$ git init
Initialized empty Git repository in .git/
```

```
local$ mkdir projekt
local$ cd projekt
local$ git init
local$ touch README
local$ git add README
local$ git commit -m "Initial commit"
local$ cd ..
local$ git clone --bare projekt projekt.git
local$ ssh -r projekt.git git@server:
local$ rm -r projekt projekt.git
local$ git clone ssh://git@server/~/projekt.git master
```

67.7.5. Public repository / git server

Odkazy:

- Setting up a public repository (http://www.kernel.org/pub/software/scm/git/docs/user-manual.html#setting-up-a-public-repository)
- Setting up a git repository which can be pushed into and pulled from over HTTP(S). (http://www.kernel.org/pub/software/scm/git/docs/howto/setup-git-server-over-http.txt)
- Git on the Server (http://progit.org/book/ch4-0.html) z knihy Pro Git
- Setting Up a Git Server (http://blog.commonthread.com/2008/4/14/setting-up-a-git-server) by Ben Wyrosdick [2008-03-13]
-

Nejdříve protokoly/způsoby kterými můžeme naše repositáře publikovat.

- Lokálně na souborovém systému. Adresář `.git` v našem projektu je tímto repositářem.
- Přístupem pomocí SSH
- Protokolem GIT.

- Přes WWW server pomocí HTTP. Můžeme publikovat repositář bez použití programu git či jeho nástrojů.

Nyní blíže k jednotlivým možnostem. Nejdříve repositář přístupný na souborovém systému. Tento samozřejmě nemusí být lokální ale může se jednat o adresář na připojeném síťovém svazku.

```
$ git clone /home/git/MujProjekt.git
$ git clone file:///home/git/MujProjekt.git

$ git clone ssh://user@server:MujProjekt.git
$ git clone user@server:MujProjekt.git
```

Existuje více způsobů, jak sprovoznit git server.

- pomocí ssh
- samostatný git server
- s pomocí http serveru apache

* *Zde by se hodilo krátké srovnání jednotlivých způsobů zveřejnění git repository. Nicméně zatím jsem žádné nenašel a nemám dost zkušeností abych je porovnal.*

67.7.5.1. Přístup do repository pomocí ssh

Odkazy:

- Howto host git on your Linux box (http://eagain.net/blog/2007/03/22/howto-host-git.html)
- gitosis README (http://eagain.net/gitweb/?p=gitosis.git;a=blob;f=README.rst)
- Setting Up a Git Server (http://blog.commonthread.com/2008/4/14/setting-up-a-git-server)
-

Na serveru si vytvoříme adresář. Já jsem si pro tento účel vytvořil rovnou uživatele abych tak oddělil přístup k repositáři od ostatních věcí.

```
# adduser --system --home /home/git --gecos "Private git repository" git
```

Abych nemusel zadávat při každé operaci s git repository heslo, nastavím si přístup přes ssh kllíč. T.j. nakopíruji na účet git na server svůj veřejný ssh klíč:

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub git@git-server.example.com
```

Vytvořenému uživateli nastavíme heslo

```
# passwd git
```

* *Git cheat sheet (http://www.gnome.org/~federico/misc/git-cheat-sheet.txt):*

```
$ git clone --bare . /tmp/myproject.git
$ git --bare --git-dir=/tmp/myproject.git update-server-info
$ chmod +x /tmp/myproject.git/hooks/post-update
$ scp -r /tmp/myproject.git ssh://user@server/~/
```

67.7.5.2. Použití gitosis

- Hosting Git repositories, The Easy (and Secure) Way (http://scie.nti.st/2007/11/14/hosting-git-repositories-the-easy-and-secure-way)

- From subversion to git (part3 - gitosis) (http://vafer.org/blog/20080115011413)
- Gitosis Cheatpage (http://archive.daniel-baumann.ch/debian/documents/cheatpages/gitosis.html)
- Gitosis (http://progit.org/book/ch4-7.html) na Pro Git
-
-
-
-

67.7.5.2.1. Instalace gitosis v Debian Lenny

> **Poznámka:** V tomto postupu je `server` počítač jenž bude sloužit jako git server a `mujpocitac` je počítač ze kterého budu gitosis administrovat a odkud budu mimo jiné využívat služeb git serveru.

Nainstalujeme balíček ze standardního repositáře.

```
server# aptitude install gitosis
```

V `/usr/share/doc/gitosis` najdeme dokumentaci, zejména pak soubory `README.Debian` `README.rst.gz` popisující co dál. Nejdříve si na server s gitosis přeneseme svůj vlastní veřejný ssh klíč.

```
mujpocitac$ scp .ssh/id_rsa.pub root@gitosis-server:/srv/gitosis/
```

Na gitosis serveru teď' inicializujeme repository.

```
server# su - gitosis
server$ gitosis-init < id_rsa.pub
```

Vše další se již odehrává z mého počítače z účtu ze kterého jsem vzal `id_rsa.pub`.

67.7.5.2.2. Instalace v Debian Etch

Balíček gitosis se nachází až v Debian Lenny. Také jsem jej nenašel v backports.org pro Etch. Použil jsem tedy jednoduchou instalaci ze zdrojů:

```
# aptitude install python python-setuptools sudo
# cd /usr/local/src
# git clone git://eagain.net/gitosis
# cd gitosis
# python setup.py install --prefix=/usr/local
```

Po instalaci je třeba zkontrolovat práva souboru `/usr/local/lib/python2.4/site-packages/gitosis-0.2-py2.4.egg/`
Tento soubor se může nacházet v jiném adresáři podle vašeho konkrétního systému a podle toho jaký jste zvolili `--prefix`. V mém případě práva na tomto souboru byla špatná, takže jsem je opravil.

```
# chmod 0755 /usr/local/lib/python2.4/site-packages/gitosis-0.2-py2.4.egg/gitosis/templates/ad
```

Když máme gitosis nainstalovaný, přikročíme k vytvoření uživatele pro repositář. Můžeme použít obvyklé jméno git neobo jakékoliv jiné. Můžeme mít také více účtů.

```
# adduser --system --home /home/git --gecos 'git repository' --group --disabled-password --she
```

Protože k repositáři budeme přistupovat přes ssh klíč, nahrajem si jej na server.

```
notebook:$ scp ~/.ssh/id_rsa.pub radek@server.cz:

server:# sudo -H -u git gitosis-init </radek/id_rsa.pub
Initialized empty Git repository in ./
Initialized empty Git repository in ./
```

S gitosis se pracuje pomocí gitu. Tedy naklonujeme si gitosis-admin a v té provádíme změny které natlačíme spátky na server. Začneme tedy klonováním.

```
notebook:$ git clone git@SERVER:gitosis-admin.git
```

Nyní můžeme přidat dalšího uživatele. To učiníme tak že do adresáře `gitosis-admin/keydir` přidáme jeho veřejný ssh klíč.

```
notebook:$ cd gitosis-admin
notebook:$ cp karel_rsa.pub keydir
notebook:$ git add keydir/karel_rsa.pub
notebook:$ git commit
notebook:$ git push
```

Vytvoření nového projektu sestává ze dvou kroků. V prvním jej zapíšeme do konfigurace a přidělíme přístup uživatelům. Konfigurace je v souboru `gitosis.conf` adresáře `gitosis-admin`. Pro nový projekt pojmenovaný pokus v něm vytvoříme sekci.

```
[group pokus]
writable = pokus
members = karel@orthrank ja@notebook
```

Změny zatlačíme na server

```
notebook:$ git commit && git push
```

Nyní si vytvoříme repositář projektu a ten také zatlačíme na server.

```
notebook:# mkdir pokus
notebook:# cd pokus
notebook:# git init
notebook:# git remote add myserver git@server.example.com:pokus.git
notebook:# vi README
notebook:# git commit ...
notebook:# git push myserver master:refs/heads/master
```

67.7.5.2.3. Administrace gitosis

* ***Attributy:*** *id="gitosis-admin"*

Administraci git serveru provádíme na dálku. Nejdříve si stáhneme konfiguraci.

```
$ cd /tmp
$ git clone gitosis@192.168.1.5:gitosis-admin.git
```

Poté založíme projekt. Ten zakládáme ve dvou fázích. Nejdrříve mu vytvoříme konfiguraci a přidělíme práva zápisu. To vše editací konfiguračního souboru `gitosis.conf` a kopírováním veřejné části ssh klíčů do adresáře `keydir`.

```
[group MujProjekt]
members = radek@dora
writable = MujProjekt
```

Provedené změny zapíšeme na server a uklidíme po sobě.

```
$ git commit -a
$ git push
$ cd
$ rm -rf /tmp/gitosis-admin
```

Nyní se přepneme do adresáře s projektem. Pokud není inicializovaný tak jej inicializujeme a po commitu jej můžeme natlačit do git serveru.

```
$ cd ~/Projekty/MujProjekt
$ git remote add muj-gitosis gitosis@192.168.1.5:MujProjekt.git
$ git push muj-gitosis master:refs/heads/master
```

67.7.5.2.4. Server

```
# git-daemon --base-path=/srv/gitosis/repositories/ --export-all
```

```
# aptitude install git-daemon-run
```

67.7.6. Různé postupy

Příkazy ro vytváření nové repository:

```
$ git-init
$ git-clone
```

67.7.6.1. Posílání patchů mailem

*

Odkazy:

- git-format-patch(1) Manual Page (https://www.kernel.org/pub/software/scm/git-core/docs/git-format-patch.html)
-

Pro tuto příležitost se používá příkaz **format-patch**.

Nejdříve si ale trochu upravíme konfiguraci. Použijeme nejlépe globální konfiguraci v souboru `~/.gitconfig`. Tento soubor upravíme přímo, nebo použijeme konfigurační příkazy gitu.

```
$ git config --global format.from  "Radek <radek@example.com>"
```

67.8. Mercurial

Odkazy:

- mercurial (http://mercurial.selenic.com/) Work easier Work faster
-
-
-

Instalace na Debian Lenny

```
# aptitude install mercurial
```

* ***FIXME:*** *doplnit, odzkoušet*

Krátké ukázky použití přímo z webu mercurial (http://mercurial.selenic.com/). Stažení (naklonování) projektu, práce na projektu a odeslání změn.

```
$ hg clone http://selenic.com/repo/hello
$ cd hello
$ # práce na projektu, editování souborů
$ hg add (new files)
$ hg commit -m 'Popis provedených změn.'
$ hg push
```

Vytvoření nového projektu.

```
$ hg init (project-directory)
$ cd (project-directory)
$ # práce na projektu, editování souborů
$ hg add
$ hg commit -m 'Initial commit'
```

Například zdrojové kódy této knihy jsem zavedl do Mercurial repository následujícím postupem.

```
$ cd ~/src/doc/book/unix        # přepnutí do adresáře projektu
~/src/doc/book/unix$ make superclean    # vyčištění od produktů překladu
~/src/doc/book/unix$ cd ..
~/src/doc/book$ hg init unix    # inicializace repositráře Mercurial
~/src/doc/book$ cd unix
~/src/doc/book/unix$ hg add     # přidání všech souborů projektu
~/src/doc/book/unix$ hg commit -m "Zavedení knihy do SCM Mercurial"
```

Pracovní cyklus projektu. Začneme tím, že máme aktualizovaný projekt. Nyní můžeme editovat jednotlivé soubory v projektu. Pokud nějaký soubor odstraníme, oznámíme to ihned příkazem

```
$ hg add soubor
```

Důležité je to udělat co nejdříve, ať na to nemusíme myslet. Pokud nějaký soubor naopak odstraníme oznámíme to příkazem

```
$ hg remove soubor
```

Průběžně se můžeme ptát jaký je stav projektu. T.j. které soubory se změnily, přibyly nebo byly odstraněny.

```
$ hg status
```

Můžeme se ptát i na rozdíly. Tedy co se v souborech změnilo. První příkaz zobrazí změny v jednom konkrétním souboru. Druhý příkaz změny ve všech souborech v projektu.

```
$ hg diff soubor
$ hg diff
```

67.9. Další systémy správy verzí

Jen lehce zmíním další programy a systémy programů jenž slouží pro správu zdorjového kódu a jeho verzí.

darcs (http://www.abridgegame.org/darcs/) — David's Advanced Revision Control System

David's Advanced Revision Control System is yet another replacement for CVS. It is written in Haskell, and has been tested on Linux and MacOS X. It has not yet been optimized for speed, so it runs a little slowly on large projects. darcs includes a cgi script, which can be used to view the contents of your repository.

From: "Sean E. Russell": By the way, I can recommend both Subversion (subversion.tigris.org) and Darcs (http://www.abridgegame.org/darcs/) as good replacements for CVS. Subversion is most orthoganal to CVS, and might be an easier migration. Darcs is better for distributed development.

Podle všeho je darcs napsno v Haskelu.

0.9.15 is the latest version, and for those who wanted more Haskell traffic, darcs is written in Haskell. Darcs' theory of patches is explained in terms of quantum mechanics for those who are interested. http://www.abridgegame.org/darcs/manual/node7.html Anyways, darcs is worth investigating. In my opinion, it has some features that make it preferable to arch, most notably simplicity and much better cherry picking of patches.

Perforce (http://www.perforce.com/index.html)

- Komerční produkt, demo je zdarma a obsahuje licenci pro dva uživatele. Pro OpenSource projekty je možno získat licenci gratis.

XP VMaster (http://sourceforge.net/projects/vmaster/)

vypadá zajímavě podle popisu. Využívá XP metodologii.

PRCS (http://sourceforge.net/projects/prcs) — the Project Revision Control System

FIXME: doplnit PRCS (http://prcs.xdelta.org/)

VESTA (http://www.vestasys.org/)

FIXME: doplnit

BitKeeper (http://www.bitkeeper.com/)

FIXME: doplnit

Bazaar-NG (http://bazaar-ng.org/)
next-generation distributed version control

Bazaar-NG (or bzr) is a project of Canonical to develop an open source distributed version control system that is powerful, friendly, and scalable. Version control means a system that keeps track of previous revisions of software source code or similar information and helps people work on it in teams.

bzr is still at a fairly early stage of development but has been self-hosting since March 2005 with no loss of data.

Mercurial (http://www.selenic.com/mercurial/wiki/index.cgi)

Mercurial: „a fast, lightweight Source Control Management system designed for efficient handling of very large distributed projects“. Mercurial is developed using Python and open source licensed.

Kapitola 68. X Window System

FIXME:TBD

Kapitola 69. Xlib

Programování aplikací pro X Window System pomocí Xlib

Odkazy:

- The X-Windows Disaster (http://catalog.com/hopkins/unix-haters/x-windows/disaster.html) from book The Unix-Haters Handbook (http://catalog.com/hopkins/unix-haters/handbook.html) by Don Hopkins (http://catalog.com/hopkins/)
- Xlib Programming Manual (http://tigr.net/afterstep/X/xlib/)
- XLib Manual (http://www.the-labs.com/X11/XLib-Manual/)
- SUPER-UX Xlib Programming Manual (http://www.hezt.net/~larson/frekko/Xlib/contents.html)
- Xlib Programming Manual (http://www.sbin.org/doc/Xlib/) (O'Reilly & Associates, Inc.
- Basic Graphics Programming With The Xlib Library (http://users.actcom.co.il/~choo/lupg/tutorials/xlib-programming/xlib-programming.html)
- Introduction to X Window programming (http://nobug.ifrance.com/nobug2/article1/babyloon/tut_xwin.htm)

69.1. Konvence jmen

Pojmenování funkcí a maker v Xlib má svůj řád. Jména funkcí vždy začínají velkým písmenenm X a používá se velkých písmen v názvu pro optické oddělení. Například:

XOpenDisplay()
XCreateSimpleWindow()

Makra a konstanty mají podobnou konstrukci jmen s tím že nezačínají velkým X:

DefaultScreen()
DisplayWidth()
GCLineStyle

Svou konvenci má i pořadí parametrů funkcí a maker. Ukazatel na strukturu Display je téměř vždy prvním parametrem.

69.2. Překlad programu

Překlad programu používajícího Xlib knihovnu.

```
$ cc prog.c -o prog -lX11
```

69.3. Kostra programu

Program vykonává ve svém životním cyklu následující operace:

1. Otevření spojení k display. (Připojení k X Window serveru.)

2. Získávání informací od serveru a zobrazovaání dat.
3. Uvolnění použité paměti
4. Uzavření spojení k serveru.

```
#include <X11/Xlib.h>  /* defines common Xlib functions adn structs. */
...
/*
 * Do proměnné display bude uložen ukazatel na strukturu Display která
 * popisuje otevřené spojení na X11 server.
 */
Display *display;

 /* Otevření spojení na display server "yoda:0.0". */
 display = XOpenDisplay("yoda:0.0");
 if (display == NULL) {
  fprintf(stderr, "Cannot connect to X server %s\n", "yoda:0.0");
  exit(-1);
 }

        /*
         * Před ukončením programu, po uvolnění všech použitých prostředků
         * uzavřeme spojení s X11 serverem.
         */
 XCloseDisplay(display);
```

69.3.1. Připojení k zobrazovacímu serveru

Prvním krokem který musí X11 program udělat je připojit se k zobrazovacímu serveru. K serveru který obsluhuje „grafický“ terminál. Pro tento účel je k dispozici funkce `XOpenDisplay`. Této funkci se předává jediný parametr, a tím je adresa zobrazovacího serveru. Tato adresa je ve formě řetězce

hostname:*server_number*.*screen_number*

Pokud je řetězec prázdný, použije se místo něj hodnota proměnné `DISPLAY` z prostředí. Celý postup připojení k X11 serveru tedy vypadá následovně.

```
#include <X11/Xlib.h>
Display *display;
char *display_name = NULL;
⋮
        if ((display = XOpenDisplay(display_name)) == NULL) {
         fprintf(stderr, "%s: cannot connect to X server %s\n",
                         argv[0], XDisplayName(display_name));
                exit(-1);
        }
```

69.4. Získání informací o zobrazovacím zařízení

Pokud máme otevřeno spojení k zobrazovacímu X11 serveru, můžem o něm získat řadu informací. Děláme tak voláním funkcí/maker knihovny xlib.

FIXME:

```
/*
 * Do této proměnné uložíme číslo standardní obrazovky X11 serveru.
 * Servery mají obvykle jednu obrazovku s číslem nula, ale mohou jich
 * mít více.
 */
int screen_num;
...
/*
 * Zjištění čísla standardní obrazovky na X11 serveru.
 */
screen_num = DefaultScreen(display);
```

Velikost obrazu X11 serveru v bodech zjistíme užitím maker `DisplayWidth` a `DisplayHeight`. Obě makra mají stejné parametry. Ukazatel na strukturu Display *`display`* a číslo obrazovky *`screen_num`*.

```
/*
 * Do těchto proměnných uložíme rozměry zobrazovacího serveru.
 */
int screen_width;               /* šířka obrazu v bodech (pixel) */
int screen_height;              /* výška obrazu v bodech (pixel) */
...
screen_width = DisplayWidth(display, screen_num);
screen_height = DisplayHeight(display, screen_num);
```

69.5. Grafický kontext

Práce s obrazem se děje v grafickém kontextu. Všechny operace které provádíme se provádějí pravě skrze grafický kontext. V tomto kontextu jsou oloženy informace které kreslící funkce používají. Mužeme se na grafický kontext dívat jako na paletu se štětcem. Paletu s pečlivě namíchanými barvami a štětec potřebného tvaru. Grafiký kontext nemusí být nutně jeden, práve naopak. Můžeme si nadefinovat řadu různých grafických kontextů, a víbírat si v kterém z nich se kreslící operace provede. Standardní kontext získáme pomocí funkce/makra **FIXME:**`DefaultGC(display, screen)`. Vlastní kontext si vytvoříme například pomocí **FIXME:**`XCreateGC(display, window, valuemask, &values)`

Pro změny v grafickém kontextu máme například funkce:

```
XSetForeground(display, gc, color);
XSetBackground(display, gc, color);
XSetFillStyle(display, gc, style);
XSetLineAttributes(display, gc, width, drawing_Style, cap_style, join_style);
```

69.6. Kreslení

Primitivní fuknce pro kreslení.

```
XDrawPoint(display, win, gc, 5, 5);
XDrawLine(display, win, gc, 20, 20, 40, 100);
XDrawArc(display, win, gc, 50-(15/2), 100-(15,2), 15, 15, 0, 360*64);
XPoints points[] = {
  { 0,  0},
  {15, 15},
```

```
    { 0, 15},
    { 0,  0}
};
int npoints = sizeov(points)/sizeof(XPoint);
XDrawLines(display, win, gc, points, npoints, CoordModeOrigin);
XFillRectangle(display, win, gc, 60, 150, 50m 60);
XFillArc(...);
XFillPolygon(...);
XFillArcs(...);
XFillRectangles(...);
```

69.7. Kreslení

Primitivní fuknce pro kreslení.

```
XDrawPoint(display, win, gc, 5, 5);
XDrawLine(display, win, gc, 20, 20, 40, 100);
XDrawArc(display, win, gc, 50-(15/2), 100-(15,2), 15, 15, 0, 360*64);
XPoints points[] = {
    { 0,  0},
    {15, 15},
    { 0, 15},
    { 0,  0}
};
int npoints = sizeov(points)/sizeof(XPoint);
XDrawLines(display, win, gc, points, npoints, CoordModeOrigin);
XFillRectangle(display, win, gc, 60, 150, 50m 60);
XFillArc(...);
XFillPolygon(...);
XFillArcs(...);
XFillRectangles(...);
```

69.8. Okna

K dispozici máme řadu funkcí pro vytváření oken. Jednou z nich je funkce **FIXME:**`XCreateSimpleWindow`. Tato funkce má 9 parametrů. Ve zkratce to jsou: *`display`*, *`parent`*, *`x`*, *`y`*, *`width`*, *`height`*, *`border_width`*, *`border`* a *`background`*. Funkce sama vraci id vytvořeného okna. Tím že okno vytvoříme ovšem ještě nedojde k jeho zobrazení. Okno musíme nejdříve "namapovat" na obrazovku funkcí **FIXME:**`XMapWindow`

```
XMapWindow(display, window);
```

A vynutit si odeslání příkazů z vyrovnávacích pamětí na X Window server příkazem **FIXME:**`XSync`. Pokud si odeslání příkazů nevynutíme, tyto by se odeslali až při naplnění vyrovnávacích pamětí.

```
XSync(display, False);
```

69.9. Obrazy a bitové mapy

Pod pojemem obrazové mapy myslím bitmapové (pixelové) obrázky.

.

69.9.1. Pixelové mapy

FIXME:TBD

Pixmap XCreatePixmap(display, d, width, height, depth

```
#
Pixmap XCreatePixmap();
```

69.9.2. Bitové mapy

FIXME:TBD

int XReadBitmapFile(display, screen, *width, *height, *bitmap, *x, *y);
int XWriteBitmapFile(display, filename, width, height, bitmap, x, y);
Pixmap XCreateBitmapFromData(display, screen, data, width, height);
Pixmap XCreateBitmapFromBitmapData(display, screen, data, width, height fg, bg, depth);

69.9.3. Obrazy (Images)

Obrazy

* ***FIXME:****TBD*

Obrazy můžeme přenášet mezi klientem a serverem pomocí funkcí `XGetImage` a `XPutImage`.

* ***FIXME:****TBD*

void `XPutImage`();
XImage *`XGetImage`();
XImage *XGetSubImage();
XImage *`XCreateImage`();
unsigned long `XGetPixel`(ximage, x, y);
void `XPutPixel`(ximage, x, y, pixel);
XImage *XSubImage();
void XAddPixel(ximage, value);
void XDestroyImage(ximage)

69.10. Přehled vybraných funkcí

DisplayHeight

Jméno

`DisplayHeight, DisplayWidth` — výška a šířka obrazovky zobrazovacího zařízení

Přehled

```
#include <X11/Xlib.h>
int DisplayHeight(Display *display, int screen);
int DisplayWidth(Display *display, int screen);
```

Argumenty

Makra mají dva parametry. Ukazatel na strukturu popisující zobrazovací zařízení *`display`* jenž byl získán voláním `XOpenDisplay`, a číslo obrazovky *`screen`* jejíž rozměry zjišťujeme. Toto číslo zase získáme voláním `DefaultScreen`.

Odkazy

Použití makra je zdokumentováno v sekci 69.4.

DefaultScreen

Jméno

`DefaultScreen` — číslo standardní obrazovky

Přehled

```
#include <X11/Xlib.h>
int DefaultScreen(Display *display);
```

Argumenty

Makra mají dva parametry. Ukazatel na strukturu popisující zobrazovací zařízení *`display`* jenž byl získán voláním `XOpenDisplay`, a číslo obrazovky *`screen`* jejíž rozměry zjišťujeme. Toto číslo zase získáme voláním `DefaultScreen`.

Odkazy

Použití makra je zdokumentováno v sekci 69.4. Související funkce: `XOpenDisplay`.

DefaultVisualOfScreen

Jméno

`DefaultVisualOfScreen` — vytvoření objektu reprezentujícího obraz

Přehled

```
#include <X11/Xlib.h>
void DefaultVisualOfScreen(XImage *ximage);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XCreateImage`.

XCreateImage

Jméno

`XCreateImage` — vytvoření objektu reprezentujícího obraz

Přehled

```
#include <X11/Xlib.h>
XImage *XCreateImage(Display *display, Visual *visual, unsigned int depth, int
format, int offset, char *data, unsigned int width, unsigned int height, int
bitmap_pad, int bytes_per_line);
```

Argumenty

display

Spojení s X serverem. Toto spojení je získáno voláním `XOpenDisplay`.

visual

Ukazatel na strukturu visual. Tuto strukturu získáme voláním `DefaultVisualOfScreen`.

depth

Barevná hloubka obrazu. Počet bitů na jeden pixel.

format

Určuje formát obrazu. Muže být XYBitmap, XYPixmap nebo ZPixmap. Tento parametr určuje jak jsou uloženy obrazová data v paměti. XYBitmap znamená jednobitové obrazy. V XYPixmap jsou uloženy jednotlivé základní barvy RGB každá zvlášt ve vlastním obraze. Data tak sestávají ze tří částí jdoucích po sobě. ZPixmap je pak obraze ve kterém jsou RGB pro jeden pixel uloženy dohromady.

offset

Počet pixelů od začátku řádku které se mají ignorovat.

data

Ukazatel na paměť s daty obrazu. Vyhrazení této paměti musí zajistit programátor.

width

Šířka obrazu v pixelech.

height

Výška obrazu v pixelech.

bitmap_pad

Specifkuje počet bitů (8, 16 nebo 32). Na toto číslo je zarovnán začátek každého řádku v obrazu.

bytes_per_line

Počet bytů mezi začátkem řádku a začátkem následujícího řádku. Nulový parametr znamená že řádky jsou uloženy bezprostředně za sebou. Celý obraz tak tvoří souvislý blok paměti.

Popis

Funkce alokuje paměť pro XImage strukturu pro zadaný displej (X server). Struktura je vyplněná podle zadaných parametrů. Funkce neprovádí alokaci paměti pro samotná obrazová `data`. Tuto paměť musí alokovat a připravit programátor před voláním funkce XCreateImage.

* ***FIXME:***

Příklad použití

```
#include <X11/Xlib.h>
⋮
        Display *display;
        Screen *screen;
        Visual *visual;
        XImage *obraz;

        display = XOpenDisplay(NULL);
        screen = DefaultScreenOfDisplay(display);
        visual = DefaultVisualOfScreen(screen);

 data = (char *)malloc(256*192*x11ui_get_display_depth);

        obraz = XCreateImage(display, DefaultVisualOfScreen(screen), x11ui_get_display_depth,
```

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.9.3. Související funkce: `XDestroyImage`, `XGetImage`, `XInitImage`, `XPutImage`.

XCreatePixmap

Jméno

`XCreatePixmap` — vytvoření pixelové mapy/matice

Přehled

```
*FIXME:#include <X11/Xlib.h>
Pixmap XCreatePixmap(Display *display, Drawable d, unsigned int width, unsigned
int height, unsigned int depth);
```

Argumenty

display

: spojení s X serverem

d

: obrazovka na X serveru

width
height

: šířka a výška pixelové matice

depth

: barevná hloubka

FIXME:

Odkazy a související funkce

Externí zdroje

- XCreatePixmap (http://tronche.com/gui/x/xlib/pixmap-and-cursor/XCreatePixmap.html) z Xlib manuálu od Chistophe Tronche

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XOpenDisplay`.

XDestroyImage

Jméno

`XDestroyImage` — vytvoření objektu reprezentujícího obraz

Přehled

```
#include <X11/Xlib.h>
void XDestroyImage(XImage *ximage);
```

Argumenty

Parametr *`ximage`* je ukazatel na strukturu XImage která byla vytvořena funkcí `XCreateImage`.

Popis

Funkce uvolní z paměti jak strukturu XImage zadanou parametrem *`ximage`*, tak také vlastní obrazová data která byla předána do struktury XImage při jejím vytváření funkcí `XCreateImage` jako parametr *`data`*.

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.9.3. Související funkce: `XCreateImage`.

XFreePixmap

Jméno

`XFreePixmap` — zrušení pixelové matice a uvolnění použitých zdrojů

Přehled

```
#include <X11/Xlib.h>
XFreePixmap(Display *display, Pixmap pixmap);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XOpenDisplay`.

XGetImage

Jméno

`XGetImage` — získání obrazu ze serveru

Přehled

```
#include <X11/Xlib.h>
XFreePixmap(Display *display, Pixmap pixmap);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XOpenDisplay`.

XGetPixel

Jméno

`XGetPixel` — získání obrazu ze serveru

Přehled

```
#include <X11/Xlib.h>
XFreePixmap(Display *display, Pixmap pixmap);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.9.3. Související funkce: `XGetPixel`, `XCreateImage`, `XGetImage`.

XInitImage

Jméno

`XInitImage` — specifikace obrazur

Přehled

```
#include <X11/Xlib.h>
Status XInitImage(XImage *image);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XGetImage`, `XPutImage`.

XOpenDisplay

Jméno

`XOpenDisplay` — připojení k X Window serveru

Přehled

```
#include <X11/Xlib.h>
Display *XOpenDisplay(char *display_name);
```

Argumenty

Parametr XOpenDisplay obsahuje řetězec popisující X11 server ke kterému se chceme připojit. Má strukturu

hostname:number.screen_number

Popis

Při úspěšném připojení vrací funkce `XOpenDisplay()` ukazatel na strukturu typu Display která opisuje otevřené připojení. Při neúspěchu vrací `NULL`.

XPutImage

Jméno

`XPutImage` — přenesení obrazu na server

Přehled

```
#include <X11/Xlib.h>
XFreePixmap(Display *display, Pixmap pixmap);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XOpenDisplay`.

XPutPixel

Jméno

`XPutPixel` — získání obrazu ze serveru

Přehled

```
#include <X11/Xlib.h>
XFreePixmap(Display *display, Pixmap pixmap);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XOpenDisplay`.

funkce šablona

Jméno

`funkce šablona` — co funkce dělá

Přehled

```
#include <X11/Xlib.h>
int funkce(Display *param1, int param2);
```

Argumenty

FIXME:

Odkazy a související funkce

Použití funkce je zdokumentováno v sekci 69.4. Související funkce: `XOpenDisplay`.

Kapitola 70. X Toolkit

```
# aptitude install libXaw7-dev
```

Kapitola 71. Tcl

* *chapter id="tcl" xreflabel="Tcl"*

Odkazy a zdroje:

- History of Tcl (http://www.tcl.tk/advocacy/tclHistory.html)
- Tcl Developer Xchange (http://www.tcl.tk)
- _ ()
- TCL základy - I. část (23.09.2004) (http://www.root.cz/print.php4?id=2413)
- TCL základy - II. část (30.09.2004) (http://www.root.cz/print.php4?id=2422)
- GUI v Tk snadno a rychle - I. část (07.10.2004) (http://www.root.cz/print.php4?id=2434)
- GUI v Tk snadno a rychle - II. část (18.11.2004) (http://www.root.cz/print.php4?id=2506)
- _ ()
- TIP #12: The "Batteries Included" Distribution (http://www.tcl.tk/cgi-bin/tct/tip/12.html)
- Tcl/Tk Aqua (http://tcltkaqua.sourceforge.net/)
- _ ()
- Připravovaná kniha Průvodce programovacími jazyky TCL/TK (http://knihy.cpress.cz/Pocitac/Book.asp?ID=1416&SearchText=Tcl&SearchType=4) od Zdislaw Wrzeszcz
- Článek z LinuxWorldu Úvod do Tcl/Tk -- časť 4: ako rýchlo programovať GUI (nielen) pod Linuxom (http://www.linuxworld.cz/lw.nsf/ID/678CFDA88E411420C1256E930058F7D2?OpenDocument&cast=1)

Tcl je jedním z dostupných skriptovacích jazyků. Jeho základy a principy fungování jsou jednoduché a přesto je to mocný nástroj v rukou administrátora či správce. Tcl je vlastně skratka pro *Tool Command Language*.

Tcl začal vyvýjet v roce 1988 John K.Osterhout. Vývoj Tk započal o rok později.

Zajímavé balíčky

- visual-tcl

71.1. Základní prvky jazyka

FIXME: [xref linkend="bash154"/]

71.1.1. Struktura souboru

První řádek jak již bývá v UNIXu zvykem je spouštěcí, tedy spouští interpret příkazů **tcl**

```
#!/usr/bin/tclsh
```

Za ním následuje text programu/scriptu. Příkazy jsou psány na samostatných řádcích, potřebujeme-li rozepsat jeden příkaz na více řádků ukončujeme je znakem \. Potřebujeme-li naopak napsat více příkazů na jeden řádek oddělíme je znakem ;. Příkazy mají tvar

```
příkaz argument1 argument2 ...
```

71.1.2. Proměnné

FIXME:

```
set proměnná hodnota
```

```
set var "Hello World"
puts $var
```

71.1.3. Příkazy

FIXME:

71.1.4. if

FIXME:

if výraz then tělo elsif výraz then tělo ... [else] tělo_N

- Klíčová slova then a else mohou být vynechána.
- Výrazem je myšlen výraz jako v expr
- Tělo může obsahovat pouze jeden příkat, výraz v hranatých závorkách nebo seznam příkazů.
- Části elseif a else nemusejí být uvedeny.

Příklad 71-1. Příklad použití příkazu if

```
set var 6
if {$var > 0} {
    puts "positive"
} elseif {$var < 0} {
    puts "negative"
} else {
    puts "zero"
}
```

71.1.5. Příkaz switch

FIXME:

switch [options] string {patern body ...}

`-exact`

Implicitní volba. Řetězec musí být shodný.

`-glob`

Při porovnání užívá stejných divokých znaků (* ? [...]) jako shell.

`-regexp`

Používá regulární výrazy.

`--`

Konec voleb.

Příklad 71-2. Příklad užití příkazu switch

```
set prom hello
switch -glob -- $prom {
  *a*   { puts "\$prom obsahuje pismeno <a>" }
  *e*   { puts "\$prom obsahuje pismeno <e>" }
  *i*   { puts "\$prom obsahuje pismeno <i>" }
  *o*   { puts "\$prom obsahuje pismeno <o>" }
  *u*   { puts "\$prom obsahuje pismeno <u>" }
}
```

71.1.6. Příkaz cyklu for

FIXME:

for start test další tělo

Příklad 71-3. Příkaz použití příkazu for

```
for { set i 1 } { $i <= 10 } { incr i } {
    puts "ahoj po $i."
}
```

71.1.7. Příkaz cyklu while

FIXME:

71.1.8. Příkaz cyklu foreach

FIXME:

71.1.9. Procedury proc

FIXME:

Příklad 71-4. Příkaz použití příkazu for

```
proc secti { x y } {
    return [expr $x + $y]
}

puts [secti 2 2]
```

71.2. Rozšíření

FIXME:

71.3. Tk (Tcl/Tk)

Odkazy a zdroje:

- Tcl/Tk Aqua (http://tcltkaqua.sourceforge.net/)
- Tcl/Tk quick start (http://www-106.ibm.com/developerworks/edu/l-dw-linuxtcl-i.html?S_TACT=104AHW03&S_CMP=EDU) ibm.com/developerWorks
- Tk Commands (http://www.tcl.tk/man/tcl8.4/TkCmd/contents.htm)
- _ ()
- _ ()

Poznámka: Předem bych rád laskavého čtenáře upozornil, že nejsem žádný Tcl/Tk Guru a v podstatě je neznám. Tyto poznámky píši při studiu Tcl/Tk a proto můj přístup může býti i zcela zcestným. Nemám nadhled jenž je třeba získat déletrvajícím aktivním používáním Tcl/Tk.

Po varování mohu tedy přistoupit k Tk. Tk je sada nástrojů pro rychlou tvorbu grafického rozhraní.

Přehled příkazů Tk: bell (http://www.tcl.tk/man/tcl8.4/TkCmd/bell.htm), bind (http://www.tcl.tk/man/tcl8.4/TkCmd/bind.htm), bindtags (http://www.tcl.tk/man/tcl8.4/TkCmd/bindtags.htm), bitmap (http://www.tcl.tk/man/tcl8.4/TkCmd/bitmap.htm), button, canvas (http://www.tcl.tk/man/tcl8.4/TkCmd/canvas.htm), checkbutton (http://www.tcl.tk/man/tcl8.4/TkCmd/checkbutton.htm), clipboard (http://www.tcl.tk/man/tcl8.4/TkCmd/clipboard.htm), colors (http://www.tcl.tk/man/tcl8.4/TkCmd/colors.htm), console (http://www.tcl.tk/man/tcl8.4/TkCmd/console.htm), cursors (http://www.tcl.tk/man/tcl8.4/TkCmd/cursors.htm), destroy (http://www.tcl.tk/man/tcl8.4/TkCmd/destroy.htm), entry, event (http://www.tcl.tk/man/tcl8.4/TkCmd/event.htm), focus (http://www.tcl.tk/man/tcl8.4/TkCmd/focus.htm), font (http://www.tcl.tk/man/tcl8.4/TkCmd/font.htm), frame (http://www.tcl.tk/man/tcl8.4/TkCmd/frame.htm), grab (http://www.tcl.tk/man/tcl8.4/TkCmd/grab.htm), grid, image (http://www.tcl.tk/man/tcl8.4/TkCmd/image.htm), keysyms (http://www.tcl.tk/man/tcl8.4/TkCmd/keysyms.htm), label, labelframe (http://www.tcl.tk/man/tcl8.4/TkCmd/labelframe.htm), listbox (http://www.tcl.tk/man/tcl8.4/TkCmd/listbox.htm), loadTk (http://www.tcl.tk/man/tcl8.4/TkCmd/loadTk.htm), lower (http://www.tcl.tk/man/tcl8.4/TkCmd/lower.htm), menu (http://www.tcl.tk/man/tcl8.4/TkCmd/menu.htm), menubutton (http://www.tcl.tk/man/tcl8.4/TkCmd/menubutton.htm), message (http://www.tcl.tk/man/tcl8.4/TkCmd/message.htm), option (http://www.tcl.tk/man/tcl8.4/TkCmd/option.htm), options (http://www.tcl.tk/man/tcl8.4/TkCmd/options.htm), pack, panedwindow (http://www.tcl.tk/man/tcl8.4/TkCmd/panedwindow.htm), photo (http://www.tcl.tk/man/tcl8.4/TkCmd/photo.htm), place, radiobutton (http://www.tcl.tk/man/tcl8.4/TkCmd/radiobutton.htm), raise (http://www.tcl.tk/man/tcl8.4/TkCmd/raise.htm), scale (http://www.tcl.tk/man/tcl8.4/TkCmd/scale.htm), scrollbar, selection (http://www.tcl.tk/man/tcl8.4/TkCmd/selection.htm), send (http://www.tcl.tk/man/tcl8.4/TkCmd/send.htm), spinbox (http://www.tcl.tk/man/tcl8.4/TkCmd/spinbox.htm), text, tk (http://www.tcl.tk/man/tcl8.4/TkCmd/tk.htm), tk_bisque (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_bisque.htm), tk_chooseColor (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_chooseColor.htm), tk_chooseDirectory (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_chooseDirectory.htm), tk_dialog (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_dialog.htm), tk_focusFollowsMouse (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_focusFollowsMouse.htm), tk_focusNext (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_focusNext.htm),

tk_focusPrev (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_focusPrev.htm), tk_getOpenFile (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_getOpenFile.htm), tk_getSavefile (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_getSavefile.htm), tk_menuSetFocus (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_menuSetFocus.htm), tk_messageBox (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_messageBox.htm), tk_optionMenu (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_optionMenu.htm), tk_popup (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_popup.htm), tk_setPalette (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_setPalette.htm), tk_textCopy (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_textCopy.htm), tk_textCut (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_textCut.htm), tk_textPaste (http://www.tcl.tk/man/tcl8.4/TkCmd/tk_textPaste.htm), tkerror (http://www.tcl.tk/man/tcl8.4/TkCmd/tkerror.htm), tkvars (http://www.tcl.tk/man/tcl8.4/TkCmd/tkvars.htm), tkwait (http://www.tcl.tk/man/tcl8.4/TkCmd/tkwait.htm), toplevel (http://www.tcl.tk/man/tcl8.4/TkCmd/toplevel.htm), winfo (http://www.tcl.tk/man/tcl8.4/TkCmd/winfo.htm), wm (http://www.tcl.tk/man/tcl8.4/TkCmd/.htm)

FIXME: ======== =========

Sada nástrojů Tk, vytvořená Johnem Ousterhoutem jako doplněk jazyka Tcl. Jedná se o knihovnu grafických objektů (widget) umožňující rychle vytvořit grafické rozhraní k aplikaci.

71.3.1. Primitiva

FIXME:

71.3.1.1. checkbutton

FIXME: zaškrtávací tlačítko

```
checkbutton .cb -text "zaskrtavaci tl." -variable have -command {puts $have}
```

71.3.1.2. radiobutton

FIXME: zaškrtávací tlačítko

```
radiobutton .rb0 -text "prepinaci tl. 1" -variable rv \
   -value "radio 1" -command {puts $rv}
```

71.3.1.3. bind

```
bind .b <Control-Button-1> {puts "Help!"}
```

71.3.1.4. frame

Rámce slouží jako kontejnery pro ostatní grafické prvky. Umožňují tím realizaci hierarchie grafických prvků.

```
#!/usr/bin/wish -f
. config -bg steelblue

foreach frame {sunken raised flat ridge groove} {
    frame .$frame -width 0.5i -height 0.5i -relief $frame -bd 2
```

```
    pack .$frame -side left -padx 10 -pady 10
}
```

71.3.1.5. toplevel

FIXME:

TopLevel primitiva jsou podobná rámcům, ale mají vlastní rodičovská okna.

```
% toplevel .n -width 2i -height 1.5i -relief ridge -bd 4
.n
```

71.3.1.6. listbox

FIXME:

71.3.1.7. canvas

FIXME:

```
% canvas .c -width 256 -height 192 -bd 2
.c
% pack .c
```

71.3.2. Widgets

71.3.2.1. button

Dokumentace: button (http://www.tcl.tk/man/tcl8.4/TkCmd/button.htm)

Prvním prvkem, který si ukážeme je tlačítko. Uveďme si tedy krátký příklad:

```
#!/usr/bin/wish -f
button .exit -text Exit -command exit
pack .exit
```

Parametry

`-text` *"text tlačítka"*

Text tlačítka, tedy nápis který se na tlačítku objeví.

`-command` *příkaz*

Příkaz který se při stisku tlačítka vykoná.

button .b

```
#!/usr/bin/wish -f
```

```
# $Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $
option add *b.activeForeground brown
pack [
      button .b -text "Hello\nWorld!!!" \
   -justify center \
   -width 20 \
   -command {puts "Hi"; exit}
]
```

71.3.2.2. entry

Dokumentace: entry (http://www.tcl.tk/man/tcl8.4/TkCmd/entry.htm)

Slouží k vytváření vstupního pole. Jedná se o klasické jednořádkové vstupní pole.

Parametry:

`-textvariable` *`proměnná`*

Propojená proměnná. Změna ve vstupním poli se projeví změnou v proměnné.

```
#!/usr/bin/wish -f
set e ""
entry .e -textvariable e
button .b -text "Do" -command {puts $e; exit}
pack .fld .btn -side left
```

`-width` *`číslo`*

Šířka vstupního pole.

`-show` *`znak`*

Znak který se má zobrazovat. Tímto způsobem můžeme realizovat pole pro vstup hesla.

```
#!/usr/bin/wish -f
set password ""
entry .pass -textvariable password -show *
button .btn -text "Do" -command {puts $password; exit}
pack .pass .btn -side left
```

Ostatní nepopsané parametry:

```
-validatecommand -vcmd
-readonlybackground
-invalidcommand

entry .e -validate key -vcmd { kontrola "%S" }
pack .e

proc kontrola {znak} {
    if {[string is integer "$znak"] == 1} {
        return true
    } else {
        puts "$znak neni cislo !"; return false
    }
}
```

71.3.2.3. label

Dokumentace: label (http://www.tcl.tk/man/tcl8.4/TkCmd/label.htm)

Příkaz label vytváří popisku. Tedy jednoduchý prvek umístněný na formuláři jenž slouží k orientaci a vlastním textem osvětluje užití jiného prvku či prvků na formuláři.

Popiska (*label*) je jednoduchý prvek obsahující text. Rozmístňujeme je na formulářích k oatatním prvkům a usnadňujeme tak orientaci uživatele.

```
% label .l1 -wraplength 2i -justify right -text "Hello World!  This is simple form."
.l1
```

% **pack .l1**

Parametry:

`-height` *číslo*

Výška popisky.

`-state` *normal/active/disabled*

FIXME:

`-width` *číslo*

Šířka popisky.

71.3.2.4. scrollbar

Dokumentace: scrollbar (http://www.tcl.tk/man/tcl8.4/TkCmd/scrollbar.htm)

FIXME:

Parametry

`option` *value*

FIXME:

71.3.2.5. text

Dokumentace: text (http://www.tcl.tk/man/tcl8.4/TkCmd/text.htm)

FIXME:

Parametry

`-text` *"text tlačítka"*

Text tlačítka, tedy nápis který se na tlačítku objeví.

71.3.3. Správci geometrie

FIXME:

71.3.3.1. pack

Dokumentace: pack (http://www.tcl.tk/man/tcl8.4/TkCmd/pack.htm)

FIXME:

Parametry:

`-side` *`left/right/top/bottom`*

Výška popisky.

71.3.3.2. grid

Dokumentace: grid (http://www.tcl.tk/man/tcl8.4/TkCmd/grid.htm)

FIXME:

```
# $Header: /home/radek/cvs/unix-book/unix.xml,v 1.7 2009-03-07 03:52:40 radek Exp $
#
# Very simple formular.

label .l_key -text "Klíč:"
entry .e_key

label .l_word -text "Slovo:"
entry .e_word


grid .l_key -row 0 -column 0 -sticky "w"
grid .e_key -row 0 -column 1 -sticky "w"
grid .l_word -row 1 -column 0 -sticky "w"
grid .e_word -row 1 -column 1 -sticky "w"
```

71.3.3.3. place

Dokumentace: place (http://www.tcl.tk/man/tcl8.4/TkCmd/place.htm)

FIXME:

71.4. Knihovny

Důležité či zajímavé knihovny.

71.5. Extrémní programování v Tcl/Tk

Odkazy a zdroje:

- TclTkUnit Project (http://park.ruru.ne.jp/ando/work/tclTkUnit/)
- TclUnit (http://sourceforge.net/projects/tclunit/) na SourceForge (http://sourceforge.net) — poslední aktualizace v roce 2002
- tcltest (http://www.tcl.tk/man/tcl8.4/TclCmd/tcltest.htm)

71.6. tclkit

Odkazy:

- Tclkit application runtime (http://www.equi4.com/tclkit/)
- Tclkit (http://wiki.tcl.tk/52)
- tclkit (http://code.google.com/p/tclkit/) on code.google.com
- Znáte tclkit? (http://www.abclinuxu.cz/blog/muf211/2006/4/znate-tclkit)
- so you want to use starkits, eh? (http://www.equi4.com/starkit/started.html)
-
-

Kapitola 72. SQLite

Odkazy:

- SQLite (http://www.sqlite.org/)
- SQLite Tutorial (http://freshmeat.net/articles/view/1428/) by Mike Chirico

SQLite je malá knihovna napsaná v jazyce C jenž realizuje jednoduchý SQL server pro malé aplikace.

72.1. Spouštěče / Triggers

* ***To be done:***

```
CREATE TRIGGER insert_t1_timeEnter AFTER INSERT ON t1
BEGIN
    UPDATE t1 SET timeEnter = DATETIME('NOW') WHERE rowid = new.rowid;
END;
```

72.2. Práce v příkazové řádce

* ***To be done:***

Prostý seznam příkazů

- **ATTACH DATABASE**, **DETACH DATABASE**
-
-
-

72.2.1. Transakce

* ***To be done:***

Kapitola 73. Programování webových aplikací v KLone

* *chapter id="klone"*

Abstrakt

KLone je programové prostředí pro vývoj webových aplikací v jazyce C/C++. Tento dokument popisuje základy použití KLone a dále praktická řešení řady problémů. KLone nám umožňuje vytvářet jednoduché a výkonné servery jako binárky. Nejsme tedy závislí na dalších prostředcích cílového systému. Při užití statického linkování dostaneme univerzálni binárku fungující na všech linuxových systémech používající stejnou třídu kompatibilních procesorů. Hlavní nasazení je však v embeded aplikacích kde jsme značně limitováni zdroji.

73.1. Předmluva

Při občasné kontrole Debian Package of the Day (http://debaday.debian.net) jsem objevil krátký článek (http://debaday.debian.net/2007/06/03/klone-c-web-programming-framework/) o KLone (http://www.koanlogic.com/kl/cont/gb/html/klone.html). KLone (http://www.koanlogic.com/kl/cont/gb/html/klone.html) je programové prostředí pro psaní webových programů v jazyce C. Článek mne natolik zaujal, že jsem se ještě ten den pokusil napsat malou aplikaci. Skončil jsem až pozdě večer, když se mi podařilo vyřešit dva problémy.

- jak přeložit celou aplikaci staticky, abych ji mohl spustit na libovolné distribuci Linuxu bez jakékoliv návaznosti na dynamické knihovny
- jak sprovoznit https v staticky překládaném programu

Proč jsem řešil takové neobvyklé problémy. Inu proto že jako prvotní použití mne napadlo jednoduché rozhraní pro spouštění několika základních úloh na velkém počtu satelitních serverů které spravuji. Servery nejsou instalovány ze stejné Linuxové distribuce a použití KLone (http://www.koanlogic.com/kl/cont/gb/html/klone.html), s šířením jediného binárního souboru mi umožní vyhnout se problémům s instalací nějakého jiného prostředí na všech servererch s řešením zvláštností jednotlivých distribucí. Popisuji zde tedy KLone právě z tohoto pohledu, tedy jak vytvářet plně samostatné binárky jenž v sobě obsahují vše potřebné bez nároků na cílový systém.

Ovšem KLone je primárně určeno pro embeded aplikace. Má vše co je pro takové aplikace třeba.

- malou velikost programu — minimální projekt má binárku velikou okolo 169987 B
- malou spotřebu paměti, kterou lze v konfiguračním souboru přesně nastavit
- výběr několika modelů serverů přesně podle potřeby aplikace (fork, prefork, iterative)

73.2. Instalace

Určitě existuje řada způsobů jak instalovat KLone. Já preferuji KLone 2.0 Bootstrap. Pro úplnnost zde však poznámky o instalaci starších verzí KLone, jakožto i instalaci z deb balíčku v Debian Etch.

73.2.1. Instalace z deb balíčku na Debian Etch

Tento způsob instalace je zmíněn v článku KLone: C web programming framework (http://debaday.debian.net/2007/06/03/klone-c-web-programming-framework/). Jeví se jednoduše a tak jsem tímto začal. Jako root vydáme příkaz

```
# apt-get install klone-package
```

Takto jsme nainstalovali verz 1.1.1 KLone (http://www.koanlogic.com/kl/cont/gb/html/klone.html) která je v Debian Etch.

73.2.2. Instalace pomocí KLone-DevKit

Instalace z balíčku v Debian Etch je sice jednoduchá, ale nainstalovaná verze není nejjednodušší a nepadařilo se mi v této instalaci přeložit aplikaci slinkovanou staticky. Po nějaké době experimentování a instalace přímo ze zdrojů jsem přišel na to že lze s výhodou použít KLone-DevKit (http://www.koanlogic.com/kl/cont/gb/html/klone-devkit.html) Stáhl jsem si tedy zdrojový balíček http://www.koanlogic.com/klone/klone-devkit-latest.tar.bz2

```
$ wget http://www.koanlogic.com/klone/klone-devkit-latest.tar.bz2
```

Stažený balíček si někam uložíme, například do `/usr/local/download`, pro opakované použití. Toto je jediná operace ke které potřebujeme rootovská práva protože jako uživatel nemáme možnost do uvedeného adresáře zapisovat.

```
# cp klone-devkit-latest.tar.bz /usr/local/download/
```

Samotná instalace se provede rozbalením zdrojového balíčku

```
$ tar xjf /usr/local/download/klone-devkit-latest.tar.bz2
```

Po rozbalení provedeme sestavení serveru ulázkové aplikace:

```
$ cd klone-devkit-1.2.1/sample-app
$ make setup
$ make
```

Jak je při provádění příkazu **make setup** vidět. Prostředí dev-kitu si stáhne z internetu zdrojové kódy KLone (http://www.koanlogic.com/kl/cont/gb/html/klone.html) a rozbalí je do ukázkové aplikace `sample-app`. Zdroje KLone (http://www.koanlogic.com/kl/cont/gb/html/klone.html) se stahují do adresáře `klone-devkit-1.2.1/dist/` dev-kitu. Při sestavování jiné ukázkové aplikace, nebo naší vlastní aplikace se používají tyto stažené zdroje.

Protože vzorovou aplikaci `sample-app` budeme používat jako šablonu, vyčistíme ji. Buď'to celé prostředí dev-kitu smažeme a rozbalíme znova, nebo v `sample-app/` spustíme příkaz **make purge**

> **Poznámka:** Jak je vidět, při tomto způsobu „instalace“ není třeba mít rootovská práva. Můžeme pracovat jen pod obyčejným uživatelem.

73.2.3. KLone 2.0 Bootstrap

Od verze 2.0 se instalace KLone přímo ze zdrojů se nedoporučuje. Doporučný postup je následující.

1. Vytvoříme si adresář v kterém bude naše aplikace. Například naše aplikace se jmenuje `dbed` a bude v adresáři `~/src/dbed`

   ```
   $ mkdir ~/src/dbed
   $ cd ~/src/dbed
   ```

2. V tomto adresáři vytvoříme `Makefile` podle následujícícho vzoru.

   ```
   KLONE_VERSION ?= 2.0.0

   # webapp content is in webapp/ the current dir
   WEBAPP_DIR = $(CURDIR)/webapp

   include klapp.mk

   klapp.mk:
           wget -O $@ -c http://koanlogic.com/klone/klapp-2.0.0.mk
   ```

   Uvedený `Makefile` je minimální. Pokud budeme potřebovat nějaká další nastavení, zde je budeme psát. Pro začátek ovšem úplně postačí.

3. V adresáři naší aplikace `~/src/dbed/` spustíme **make**.

   ```
   $ cd ~/src/dbed
   $ make
   ```

   Tento příkaz stáhne celé vývojové prostředí i se zdroji jednoduché aplikace "Hello World". Tato slouží jako šablona pro další vývoj a samozřejme také pro ověření že nám vše funguje jak se přesvědčíme v následujícím kroku.

4. Vzniklý program **kloned** s naší aplikací si odzkoušíme.

   ```
   $ ./kloned -F
   $ firefox http://localhost:8080
   ```

   Přepínač `-F` zajistí že démon kloned se spouští na popředí. Prohlížeč tedy spouštíme z jiného terminálu, nebo již spuštěnému prohlížeči zadáme url `http://localhost:8080`

5. V této chvíli máme stažené celé prostředí a přeloženou vzorovou aplikaci. Můžeme tedy započít s vývojem upravováním této vzorové aplikace. Její zdroje jsou v adresáří `webapp/www`. V našem konkrétním případě tedy v `~/src/dbed/webapp/www`

73.3. Vytvoření jednoduché aplikace s použitím dev-kitu

Nejdříve si ukážeme jak vytvořit jednoduchou aplikaci s použitím dev-kitu.

Přepneme se do adresáře kam jsem dev-kit rozbalili, v mém případě do `~/src/klone-devkit-1.2.1` a vytvoříme v něm nový podadresář jako kopii adresáře sample-app.

```
$ cd ~/src/klone-devkit-1.2.1
$ cp -a sample-app first-app
```

V tuto chvíli máme funkční kostru minimalistické aplikace. Předtím než v ní začneme provádět změny ji přeložíme. Děláme to proto, abychom se přesvědčili že nikde není problém a jsme schopni úspěšně překládat.

```
$ cd first-app
$ make setup
```

```
$ make
```

V adresáři naší `first-app` se objevili dva nové soubory: `first-appd` a `first-appd-stripped`. Toto je přeložený program a jeho „odlehčená“ verze zbavená ladicích informací. Program spustíme, a webovým prohlížečem se přesvědčíme že opravdu funguje.

```
$ first-appd -F
$ firefox localhost:8080
```

Druhý příkaz zadáváme z jiné konzoly (terminálového okna).

Nyní provedeme úpravy, tedy začneme psát vlastní aplikaci upravováním šablony/příkladu. V této chvíli se soustředíme na adresář `embfs/www/` aplikace. Zde se nacházejí soubory, které budou přeloženy do aplikace. Zde vytváříme čisté html stránky (v souborech `.html`) a programované stránky (v souborech s příponou `.kl1` nebo `.klone`).

Poznámka: Pokud do `embfs/` přidáme či odtud odebereme jakýkoliv soubor, musíme před vlastním sestavením programu přikazem **make** provést přikaz **make import**.

73.4. Vlákna / Thrading

* ***To be done:***

KLone umožňuje použití tří různých modelů fungování serveru. Jsou to

- prefork — standardní model, procesy jsou spouštěny dopředu „do zásoby“, větší výkon, rachlejší odezva
- fork — úsporný model, procesy jsou spouštěny až když je jich třeba, nižší výkon, pomalejší odazva
- iterative — klienti jsou obsluhováni jen jedním procesem

V módu *prefork* je při spuštění odstartováno několik serverů „do zásoby“, které čekají na dotazy od klientů. Tento model poskytuje největší výkon a nejrychlejší odezvu.

V módu *fork* se spouští nové servery až při příchodu doatzu od klientů. Nemáme tedy žádné servery spuštěné „do zásoby“. Tento model je úsporný k prostředkům a je určen pro malý síťový provoz. Má delší dobu odezvy než předchozí model *prefork*.

Poslední mód *iterative* používá jen jeden proces. To znamená že nedochází k žádnému paralelnímu vyřizování žádostí jednotlivými procesy jako u obou předchozích modelů.

Výběr mezi model se provádí v konfiguračním souboru

```
server_list app_http
...
app_http
{
        type            http
        model           iterative

        addr.type       IPv4
        addr.port       8080
        dir_root        /www
}
```

73.5. Komponenty

* ***To be done:***

  Komponenta je samostatný kl1 soubor.

73.6. Šablony / Templates

* ***To be done:***

  Šablonu stránky realizuji ve dvou samostatných kl1 souborech. Jeden obsahuje kód stránky před obsahem a druhý kód stránky po obsahu.

```
<%!
%><%
  ... Nastavení hodnot které použijeme ve stránce nebo šabloně.
  title = "Moje stránka"
%>
<%@include ../../components/pre_content.kl1%>
  ⋮ obsah stránky
<%@include ../components/post_content.kl1%>
```

73.7. Ladění

Chceme li aplikaci ladit, musíme ji nejdříve přeložit s podporou ladění. To znamená že v hlavním souboru `Makefile` přídáme do proměnné `KLONE_CONF_ARGS` přepínač `--enable_debug --enable_debug`. Nyní když máme zajištěno že v přeloženém programu bude podpora ladění tuto použijeme. Máme k tomu dva odlišné nástroje. Prvním a jednodušším (záleží jak pro koho) jsou ladicí výpisy. Ty se zapisují do deníku.

```
#include <u/libu.h>
dbg_if(int value);
```

* ***To be done:***

73.8. Řešené problémy

Zde uvádím samostatně popisy a řešení problémů se kterými jsem se setkal.

73.8.1. Přeložení do staticky slinkované binárky

Jak jsem se zmínil v úvodu, toto první problém který jsem řešil. Tedy jak přeložíme aplikaci tak, aby běžela na „libovolné“ distribuci linuxu bez nutnosti ji pokaždé znovu překládat. Tedy vytvoření jedné binárky spustitelné všude.

Poznamenávám že podmínkou je stejná architektura. Binární program tedy překládáme pro procesory Intel a neočekávejte že takto přeložený program poběží na jiných procesorech jako jsou třeba PowerPC, MIPS či ARM.

Takže zase spátky k našemu problému. Přeložit program a slinkovat staticky je velmi jednoduché. Jediné co potřebujeme je přidat volbu `-static` na správné místo v souboru `Makefile-webapp`. Volbu zapíšeme do `WEBAPP-LDFLAGS`

```
# dynamically linked libraries
WEBAPP_LDFLAGS = -static

$ make clean
$ make setup
$ make
```

Poznámka: Pokud se nám při překladu objevují chyby jako:

```
(.text+0x72a): undefined reference to ‘dlerror’
```

Máme v systému nainstalovánu vývojářskou verzi knihovny `libssl-dev`. Přidáme ještě k volbě `-static` volbu `-ldl` jak je popsáno v 73.8.2.

73.8.2. Https v staticky sestavené aplikaci

Pokud potřebujeme používat protokol HTTPS v staticky sestavené aplikaci, musíme mít v systému nainstalovánu knihovnu `libssl-dev` a do souboru `Makefile-webapp` přidat volbu `-ldl`.

```
# dynamically linked libraries
WEBAPP_LDFLAGS = -ldl -static

# aptitude install libssl-dev
$ make clean
$ make setup
$ make
```

73.8.3. Statická html stránka

Součástí naší aplikace mohou být statické html stránky. Takovou stránku jako soubor uložíme do adresáře `embfs/www/`. Uvádím příklad takovéto stránky `static.html`:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
<html>
    <title>Statická stránka</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  </head>

  <body>
    <h1>Statická stránka aplikace <tt>fist-app</tt></h1>

    <p>Tato stránka obsahuje statický obsah.  Tedy obsah který se nemění.</p>
  </body>
</html>
```

Poté oznámíme dev-kitu že jsme vytvořili nový soubor v `embfs/www/`, a aplikaci přeložíme.

```
$ make import
$ make
```

73.8.4. Použití SQLite3

Předpokládám, že máme nainstalovánu vývojovou verzi SQLite s hlavičkovými soubory. Pokud ne, nainstalujeme ji. Na Debian Etch to bude příkazem

```
# aptitude install libsqlite3-dev
```

Upravíme `Makefile` tak, že do `WEBAPP_LDFLAGS` přidáme knihovnu sqlite3

```
WEBAPP_LDFLAGS = -lsqlite3
```

Jednoduchý příklad, spíše kostra stránky používající přístup k databázi.

```
<%!
#include <sqlite3.h>

#define MYDB "moje-databaze"
sqlite3 *db;                    /* database handler */

int callback(void *NotUsed, int argc, char **argv, char **azColName)
{
        io_printf(out, ...);
}
%>
<html>
  <head>
    ...
  </head>
  <body>
  <%
        sqlite3_open(MYDB, &db);
 ⋮
 sqlite3_exec(db, "SELECT * FROM tabulka ORDER BY surname", callback, ...);
        ⋮
        sqlite3_close(db);
  %>
  </body>
</html>
```

73.8.4.1. Obsah tabulky jako tabulka

Následující ukázka zobrazí obsah tabulky v tabulkové formě s užitím html tagu `table`.

```
<%!
    #include <sqlite3.h>
    #define MYDB "moje-databaze"
    #define SQL  "SELECT * FROM lidi"

    sqlite3 *db;
```

```
    int     retval;
    char    *zErrMsg = 0;
    int     first_line = 1;

    int callback(void *NotUsed, int argc, char **argv, char **azColName)
    {
        int     i;
        io_printf(out, "<tr>");
        for (i = 0; i < argc; i++) {
            if (first_line)
                io_printf(out, "<th>%s</th>", azColName[i]);
            else
                io_printf(out, "<td>%s</td>", argv[i]);
        }
        io_printf(out, "</tr>");
        first_line = 0;
    }
%>
<html>
  <head>
    <title>Lidi</title>
    <link rel="stylesheet" href="kl.css" type="text/css" />
    ...
  </head>
  <body>
    <table rules="all" border="1">
    <%
      retval = sqlite3_open(MYDB, &db);
      retval = sqlite3_exec(db, SQL " LIMIT 1", callback, 0, &zErrMsg);
      retval = sqlite3_exec(db, SQL           , callback, 0, &zErrMsg);
      sqlite3_close(db);
    %>
    </table>
  </body>
</html>
```

Program je značně zjednodušený ale ukazuje princip. Co stojí za povšimnutí je způsob zobrazení názvů sloupců. Protože jsem nenašel jiný, dělám to takto. Na začátku stránky se nastaví proměnná `first_line` na hodnotu 1 jenž znamená že ještě nebyl zpracován první řádek. Když zobrazuji data z databáze pomocí SQL SELECT příkazu, provedu operaci dvakrát. V prvním případě přidám do SQL příkazu klauzuli `LIMIT 1` která omezí dotaz jen na jeden výsledný řádek. S tímto řádkem je volána fukce `callback`. Ta podle obsahu proměnné `first_line` zjistí že je třeba tisknout hlavičky tabulky a zajistí to. Po zpracování jakéhokoliv řádku, tedy hlavně hlaviček se změní hodnota `first_line` na `0` a zajistí že přinásledujícím volání již budou zobrazována vlastní data. K tomu dojde po druhém volání `sqlite3_exec` tentokráte s selectem bez klauzule `LIMIT 1`.

Jiným způsobem jak rozlišit mezi hlavičkou a daty je využít prvního parametru předávaného funkci `callback`. Funkce se mírně upraví, aby místo `first_line` používala první argument *`arg1`*.

```
int callback(void *arg1, int argc, char **argv, char **azColName)
{
        ...
        if ((int) arg1))
                io_printf(out, "<th>%s</th>", azColName[i]);
            else
                io_printf(out, "<td>%s</td>", argv[i]);
        ...
}
```

A v zobrazení stránky použitím parametru `callback` funkce rozlišíme jestli se zobrazuje řádek hlaviček nebo řádek dat.

```
...
retval = sqlite3_open(MYDB, &db);
retval = sqlite3_exec(db, SQL " LIMIT 1", callback, (void *)1, &zErrMsg);
retval = sqlite3_exec(db, SQL           , callback, (void *)0, &zErrMsg);
sqlite3_close(db);
...
```

V příkladu nejsou ošetřeny všechny chybové stavy, a lze udělat řadu vylepšení. Například před zobrazením tabulky testovat jsou-li v databázi vůbec nějaké řádky, a když ne zobrazit místo tabulky informující zprávu. Je možné také s využitím CSS zobrazit tabulku proužkovaně světlejšími a tmavšími proužky.

73.8.5. Přihlašování

Jeden z úkolů který potřebujeme často řešit je přihlašování do webu. Přihlašování jsem zatím řešil na jednoduchém principu. Ten spočívá v kontrole uživatelského jména v sezení (session). Není-li v sezení definováno uživatelské jméno (username), přesměruji klienta na přihlašovací stránku login.kl1.

Ve stránce která je chráněna uvedu tedy následující od kontrolující přítomnost uživatelského jména v session.

```
<%

/* Check if user is logged in.  Redirect to login if otherwise. */
username = session_get(session, "username");
if (!username) {
 response_redirect(response, "login.kl1");
}

%>
```

Stránka pro zadávání jména a hesla je v souboru `login.kl1`.

```
<%!
/*
 * Přihlašovací formulář.
 */

static char *username = NULL;
static char *password = NULL;

%><%
/* get list of arguments */
vars_t *args = request_get_args(request);
username = vars_get_value(args, "username");
%>
<html>
  <head>
    <title>Login</title>
    <link rel="stylesheet" href="kl.css" type="text/css" />
    <meta http-equiv="content-type" content="text/html;charset=UTF-8" />
  </head>

  <body>
```

```
        <h1>Login</h1>
        <%

    switch (vars_get_value_i(args, "action"))
    {
    case 1:
     username = vars_get_value(args, "username");
     password = vars_get_value(args, "password");

     if (username && strlen(username) && password && strlen(password)) {
      /* authenticate here */
      session_set(session, "username", username);
      response_redirect(response, "/index.kl1");

      io_printf(out, "Do something with (username, password):" \
              " (%s,%s)", username, password);
     } else {
      io_printf(out, "Bad (username, passwprd)");
     }
    }
    %>

        <form action="<%= SCRIPT_NAME %>" method="post">
          <table rules="all" border="1">
            <tr>
              <th>username:</th>
              <td><input type="text" name="username"/></td>
            </tr>
            <tr>
              <th>password:</th>
              <td><input type="password" name="password"/></td>
            </tr>
            <tr>
              <td colspan="2"><input type="submit" value="login"/></td>
            </tr>
          </table>
          <input type="hidden" value="1" name="action"/>
        </form>

      </body>
    </html>
```

* ***To be done:***

73.8.6. Přihlašování znovu a lépe

Popis jakým způsobem implementovat web jehož části jsou chráněné přihlašovací procedurou.

Princip fungování je následující. Stránka, která je chráněna, si ověří, zdali je uživatel přihlášen. Jestli ano, zobrazí se normálně. Není-li uživatel přihlášen provede se přesměrování na přihlašovací stránku.

```
response_redirect(response, "login.kl1");
```

Jak poznáme že je uživatel přihlášen? Podle nastavení proměnné v session:

```
const char *username = session_get(session, "username");
if (username == NULL) {
    /* Uživatel není přihlášen */
    response_redirect(response, "login.kl1");
    return;
}
```

Samotné přihlášení je implementováno ve stránkách `login.kl1` a `authenticate.kl1`. V `login.kl1` zobrazíme jen formulář pro zadání uživatelského jména a hesla:

```
<html>
  ⋮
    <form action="authenticate.kl1" method="post">
        Username: <input type="text" name="username"><br/>
        Password: <input type="password" name="password"><br/>
        <input type="submit" name="login">
    </form>
  ⋮
</html>
```

A samotné ověření užovatele v `authenticate.kl1` vypadá takto:

```
vars_t *args = request_get_args(request);
const char *username = NULL;
const char *password = NULL;

if (vars_get_value(args, "login")) {
        username = vars_get_value(args, "username");
        password = vars_get_value(args, "password");

 if (password_match(username, password) {
                session_set(session, "username", username);
        } else {
                session_del(session, "username");
        }
}
```

Zde použitá funkce `password_match` provádí ověření jména a hesla. Může vypadat nějak takto:

```
int password_match(const char *username, const char *password)
{
        if ((strcmp("radek", username) == 0) && (strcmp("heslo", password) == 0)) return -1;
        if ((strcmp("pavel", username) == 0) && (strcmp("pomeranc", password) == 0)) return -1
 return 0
}
```

Pokud potřebujeme uživatele odhlásit, provádí se to odstraněním jeho uživatelského jména se session. Tedy jedná se o stejný příkaz který jsme použili v `authenticate.kl1` v případě že jméno neodpovídá heslu.

session_del(session, "username");

Toto je v základu princip jak přihlášení a odhlášení funguje. Při rozpracování na konkrétní případ může kód narůst o spoustu dalších věcí. Například samostatnou stránku `authenticate.kl1` můžeme zakomponovat do stránky `login.kl1`.

```
<%!  webapp/www/login.kl1
static char *username = NULL;
```

```
    static char *password = NULL;
    %><%
    vars_t *args = request_get_args(request);

    /*
     * Podle přítomnosti proměnné formuláře "login" poznáme, zdali přicházíme
     * s vyplněným username a password z tohoto formuláře.
     */
    if (vars_get_value(args, "login")) {
            username = vars_get_value(args, "username");
            password = vars_get_value(args, "password");
            if (password_match(username, password)) {
                    warn_if (session_set(session, "username", username));

      /*
                     * Přesměrování na původní chráněnou stránku, jestli byla zadána.
                     *  Jinak na index.kl1
                     */
                    redirector = session_get(session, "redirector");
                    warn_if (response_redirect(response, redirector ? redirector : "index.kl1"));
            } else {
             warn_if (session_del(session, "username"));
            }
    }
    %>
    <html>
      <head>
        <title>Login</title>
        ...
      </head>
      <body>
        <form action="login" method="post">
          Username: <input type="text" name="username"/><br/>
          Password: <input type="password" name="password"/><br/>
          <input type="submit", value="login"/>
        </form>
      </body>
    </html>
```

73.8.7. Formulář s tabulkou

Jedním z běžných problémů, se kterým se potkávám, je zobrazený dat v tabulkové formě. Například z databáze. Každý řáádek tabulky reprezentuje jeden řádek v databázové tabulce.

V kódu zobrazíme tabulku podle vzoru:

```
    <form>
      <table>
        <caption></caption>
        <thead>
          <tr>...</tr>
        </thead>
        <tfoot>
          <tr>...</tr>
        </tfoot>
        <tbody>
```

```
      <tr>...</tr>
    </tbody>
    </table>
</form>
```

Část textu tabulky bude zakoódována přímo v programu, ale části, zejména ta s daty v tělě, budeme určitě vytvářet programově. Myšlenka práce s tabulkou je taková, že v tabulce rozmístníme vstupní pole a tlačítka, podlo toho co chcem uživateli nabídnout za akce. Většinou to dělám tak, že poslední, úplně pravý, sloupec vyhradím pro příkazy a dávám do něj tlačítka pro jednotlivé akce které se týkají řádku tabulky. Na konec tabulky, ve vzoru je to část `<tfoot>`, vložím pole pro zadávání hodnoty a v poslední buňce, která je vlastně ve sloupečku pro příkazy dám tlačítko „Zapsat“. Tolik tedy na úvod a nyní se začneme zabývat jednotlivými částmi.

První věc je samotný formulář. Jak je vidět na ukázce, tabulka je ve formuláři celá, a všechny prvky které budeme dávat do tabulky budou tedy v jednom formuláři. Formulář může obsahovat ve svém otevíracím tagu několik atributů. Zmíním zde jen `url` a `method`. Attribut `url` obsahuje adresu na kterou se data z formuláře odešlou při stisknutí odesílacího, nebo některého z více odesílacích tlačítek. Pokud nebude vyplněn, odešlou se data na tu samou stránku ve které je formulář. Z našeho pohledu to znamená že data z formuláře bude zpracovávat ta samá `.kl1` stránka. Atribut `method` pak určuje jakým způsobem se budou data odesílat. V mém případě jsem použil tuto kombinaci:

```
<form method="post">
    ...
</form>
```

Tedy data fe formuláři se pošlou ke zpracování té samé stránce (`.kl1`), která formulář zobrazuje a k odeslání se použije metoda POST.

Dále do formuláře/tabulky rozmístníme jednotlivá pole. Uvedu jako příklad pole v posledním řádku tabulky pro zadání hodnot k vytvoření nového řádku.

```
<tfoot><tr>
    <td><input type="text" id="dny" size="3" maxlength="3" /></td>
    <td><input type="text" id="limit" size="3" maxlength="3" /></td>
    <td><button name="tlacitko" value="new">Zapsat</button></td>
</tr></tfoot>
```

V této ukázce jsou dve políčka pro zadání hodnoty a ve sloupci s příkazy je tlačítko jenž má nápis „Zapsat“. Políčka jsou vytvářena html tagem `<input type="text">` a můžeme a zajisté vaužijeme také ostatních variant tagu input které nám HTML nabízí. Můžeme také použít tag select pro vytvoření komboboxu s výběrem hondot. Zvláštností je odesílací tlačítko které generuji, v tomto případě možná zbytečne, tagem `<button>`. Běžne se používá kód

```
<td><input type="submit" value="Zapsat" /></td>
```

V tomto případě si mohu vybrat, který způsob použiji. V příkazových tlačítcích jenž jsou na každém řádku ovšem jde použít jen varianta s tagem `<button>`.

A co že je tedy v těch řádcích, tam generuji html kód jako tento:

```
<td>
  <button name="remove" value="klíč-řádku">Odstranit</button>
  <button name="edit"   value="klíč-řádku">Upravit</button>
</td>
```

Klíč-řádku je hodnota, která jednoznačně identifikuje data na řádku. Může to být například `id` pole z databáze, nebo jen číslo řádku tabulky. Atribut `name` pak identifikuje operaci kterou s daným řádkem chceme provést. Tím máme popsána vstupní pole formuláře a tlačítka, a zbývá nám zajistit zpracování dat z odeslaného formuláře.

```
if (request_get_arg(request, "new")) {
    ...  Uživatel stiskl tlačítko „Zapsat“ na poslendím řádku tabulky.
    ...  Vybereme tedy z request pole dny a limit a zapíšeme například do databáze.
} else if (key=request_get_arg(request, "remove")) {
    ...  Uživatel stisk tlačítko odstranit na řádku s daty jehož identifikace je v key.
    ...  Odstraníme tento řádek z databáze.
} else if (key=request_get_arg(request, "edit")) {
    ...  Uživatel stisk tlačítko „edit“ na řádku s daty jehož identifikace je v key.
    ...  Zajistíme tedy zobrazení formuláře s daty jednoho řádku.
}
```

73.9. Poznámky k funkcím

73.9.1. Databáze hesel

- u_pwd_init()
- u_pwd_term()
- u_pwd_auth_user(g_pwd, const char *usr, const char *pwd)
- u_pwd_retr(const char *pwd, const char *user, &user_rec)
- u_pwd_rec_get_opaque(user_rec)
-

Od verze 2.1.0 obsahuje KLone funkce pro práci s databází hesel. Databáze má jednoduchou strukturu.

name:md5hash:rest

Databáze je uložena v souboru, buď to v samotné aplikaci, nebo ji lze přesměrovat na soubor na filesystému. Nejdříve soubor v aplikaci. Funkce pracující s touto databází předpokládají že ji máme otevřenou a proto jim ji musíme připravit. Tak učiníme pomocí hooks kdy zajistíme otevření souboru při spuštění aplikace.

Příklad 73-1. Soubor `auth_hooks.c` z ukázkové aplikace `klapp-auth-simple-1.1.0`

```
#include <klone/klone.h>
#include <u/libu.h>

u_pwd_t *g_pwd = NULL;

static char *embfs_fgets (char *str, int size, void *stream);

int auth_init(void)
{
        io_t *pwd_io;
        const char *pwd_name = "/etc/passwd";

        /*
         * dbg_err_if (emb_open(pwd_name, &pwd_io));
         * dbg_err_if (u_pwd_init(pwd_io, embfs_fgets, , &g_pwd))
         */
```

```
 dbg_err_if (u_pwd_init_agnostic(pwd_name, u_md5, MD5_DIGEST_BUFSZ, &g_pwd));
        return 0;
err:
        return ~0;
}

int auth_term(void)
{
        return u_pwd_term(g_pwd);
}

void hooks_setup(void)
{
        hook_server_init(auth_init);
        hook_server_term(auth_term);
        return;
}

static char *embfs_fgets (char *str, int size, void *stream)
{
        io_t *io = (io_t *) stream;
        dbg_err_if (io_gets(io, str, size) <= 0);
        return str;
err:
        return NULL;
}
```

K uvedenému souboru který je v samostatném adresáři `auth_hooks` patří `Makefile`

```
# always include common.mk in MaKL-based makefiles
include common.mk

# include klone-related variables
include ../vars.mk

# name of the library
LIB = auth-hooks

# list of source files
SRCS = auth_hooks.c

# compilation flags
CFLAGS += -I ../include $(KLONE_CFLAGS)

include lib.mk
```

Hesla si zahašujeme například pomocí ruby a zapíšeme do souboru.

```
$ irb
irb> require 'digest/md5'
irb> Digest::MD5.hexdigest('stewy')
```

if you need to add some "policy" to your master password file you can use the .opaque field of an u_pwd_rec_t, pushing the intended semantics (i.e. time limits, usage counter, bad logins counter, etc.) into it:

```
user:db33f30c28364cd44195ed6105b82c29:my_policy_attrs

{
        ...
        dbg_err_if(u_pwd_retr(pwd, user, &user_rec));
        user_policy_attr = u_pwd_rec_get_opaque(user_rec);
        if (check_my_policy(user_policy_attr))
                ...
}
```

Aby se hooks uplatnili, přidáme do hlavního `Makefile` řádky

```
KLONE_CONF_ARGS += --enable_hooks
WEBAPP_LDADD = $(CURDIR)/auth-hooks/libauth-hooks.a
```

První oznamuje, že se mají povolit hooks, druhý pak říká že se má k aplikaci přidat knihovna v níž je soubor `auth_hooks.c`

Pokud chceme mít soubor s hesly jinde než zakompilovaný do aplikace, provedeme následující úpravy:

- Do hlavního `Makefile` přidáme

  ```
  KLONE_CONF_ARGS += --enable_sup_fs
  ```

-
-
-

73.10. Klone 2.1.1rc9

Kapitola 74. Java

Programovací jazyk Java, jeho prostředí knihovny a základy programování.

74.1.

74.1.1. Netříděné poznámky

Java je na blackdown:

- `j2re1.3` — Blackdown Java™
- `j2sdk1.3-demo` — Blackdown Java™
- `j2sdk1.3-src` — Blackdown Java™
- ...

Pro Debian GNU/Linux Woody jsou k dispozici zdroje

```
deb ftp://ftp.tux.org/pub/java/debian woody main non-free
```

Balíčky jsou taky na http://z42.de/debian/

74.1.2. Nasazení Javy v Debian Sarge

* *section id="java.sarge"*

Java v Sarge není, z licenčních důvodů, zato máme k dispozici balíček `java-package` který nám má být naápomocen v jejím nasazaní. Nainstalujem jej tedy a začteme se do dokumentace.

```
# aptitude install java-package
```

Jak se praví v dokumentaci, a tady ji budu opakovat, přináší nám balíček program **make-jpkg**. Tento nám dovoluje **FIXME:**nainstalovat některou z

- Sun Java 1.4 a 1.5
- Blackdown Java J2SE 1.3 až 1.4.2
- IBM J2SE 1.3.1 až 1.4.2

Pro první pokus jsem zvolil implementaci od Sunu. Nejdříve je nutno si od Sunu stáhnout balíček s Javou. Stáhl jsem nejnovější co byl k dispozici (ke dni 2005-07-09), JDK 5.0 Update 4 soubor `jdk-1_5_0_04-linux-i586.bin`

```
radek@yoda:~/tmp: 1 $ fakeroot make-jpkg jdk-1_5_0_04-linux-i586.bin
Creating temporary directory: /tmp/make-jpkg.XXXX5FZxxd
Loading plugins: blackdown-j2re.sh blackdown-j2sdk.sh common.sh ibm-j2re.sh ibm-j2sdk.sh j2re.

Detected product:
    Java(TM) Software Development Kit (J2SDK)
    Standard Edition, Version 1.5.0+update04
    Sun Microsystems(TM), Inc.
Is this correct [Y/n]:
```

```
Checking free diskspace: done.

Please enter your full name. This value will be used in the maintainer
field of the created package.

Full name [root]:Radek Hnilica

Please enter a valid email address or press return to accept the
default value. This address will be used in the maintainer field of
the created package.

Email [root@yoda]: nospam@hnilica.cz

In the next step, the binary file will be extracted. Probably a
license agreement will be displayed. Please read this agreement
carefully. If you do not agree to the displayed license terms, the
package will not be built.

...

The Debian package has been created in the current directory. You can
install the package as root (e.g. dpkg -i sun-j2sdk1.5_1.5.0+update04_i386.deb).


Removing temporary directory: done
radek@yoda:~/tmp: 0 $
```

Vytvořený balíček nainstalujeme. Já používám svoji vlastní *repository* pro lokální balíčky.

```
# cp /home/radek/tmp/sun-j2sdk1.5*.deb /root/debs
# update-debs
# aptitude update
# aptitude install sun-j2sdk1.5
# update-alternatives --config java
```

Po instalaci a konfiguraci se přesvědčíme že je vše v pořádku. Podíváme se na verzi javy a odzkoušíme aplikace které javu používají.

```
# java -showversion
java version "1.5.0_04"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_04-b05)
Java HotSpot(TM) Client VM (build 1.5.0_04-b05, mixed mode, sharing)
...
```

74.2. Tutoriály a výukové materiály

Než uvedu další informace o Javě, chtěl bych se chvíli věnovat materiálům, které jsou na internetu k dispozici.

Nezpracované informace:

- John’s Java (http://johnsjava.webs.com/)
- soulisland Java tutorials (http://www.youtube.com/user/soulisland#p/c/FF09198E3A32D630) na YouTube

74.2.1. Videotutoriály na YouTube

Na YouTube (http://www.youtube.com) jsem našel několik tutoriálů různéhoho rozsahu, kvality a zaměření. Zde některé uvedu.

74.2.1.1. Gorilla3D

Odkazy:

- Gorilla3D (http://www.gorilla3d.com/v8/index.php) Design, Develop, Deploy
- Gorilla3D (http://www.youtube.com/user/gorilla3d) YouTube profile

Uživatel Gorilla3D (http://www.youtube.com/user/gorilla3d) uveřejnil v době od 12. května 2007 do 29. srpna téhož roku sérii krátkých videí věnovaných programování v Javě. Jako prostředí v ukázkách používá NetBeans.

* *Java 1.6 with NetBeans 5.5.1 beta*

Videa:

- Java 01: Hello World (http://www.youtube.com/watch?v=J0Rc6aMEL-Y) (zveřejněno 2007-05-12)
- Java 02: Variables (http://www.youtube.com/watch?v=tIz6QXJC6pI) (zveřejněno 2007-05-12)
- Java 03: Functions (http://www.youtube.com/watch?v=wJd9_gdeij8) (zveřejněno 2007-05-13)
- Java 04: If & Switch Statements (http://www.youtube.com/watch?v=gUCYTuHXcnE) (zveřejněno 2007-05-13)
- Java 05: While & For Loops Statements (http://www.youtube.com/watch?v=m10TuI3adxw) (zveřejněno 2007-05-13)
- Java 06: Basic Classes (http://www.youtube.com/watch?v=0wh_O2v3cD8) (zveřejněno 2007-05-13)
- Java 07: Namespace & User Input (http://www.youtube.com/watch?v=Y1uacuNmuk0) (zveřejněno 2007-05-13)
- Java 08: Advanced Arrays (http://www.youtube.com/watch?v=_k-v0l2A6hQ) (zveřejněno 2007-05-13)
- Java 09: Inheritence, Abstract Classes & Methods (http://www.youtube.com/watch?v=_5xLfuMtOeE) (zveřejněno 2007-05-13)
- Java 10: Interfaces (http://www.youtube.com/watch?v=YjBAgcTOJ7M) (zveřejněno 2007-05-13)
- Java 11: Swing GUI Intro (http://www.youtube.com/watch?v=mYRSs--7E14) (zveřejněno 2007-08-29)

74.2.1.2. thenewboston

Odkazy:

- Welcome to the Land of thenewboston (http://thenewboston.com/) (v říjnu 2009 stále nedokončené webové stránky)
- thenewboston (http://www.youtube.com/user/thenewboston) YouTube profile
- YouTube playlist Java Programming Tutorials (http://www.youtube.com/view_play_list?p=FE2CE09D83EE3E28)

Uživatel thenewboston (http://www.youtube.com/user/thenewboston) uveřejňuje od 2. května 2009 na YouTube tutoriál o programování v Javě (http://www.youtube.com/view_play_list?p=FE2CE09D83EE3E28). Jako prostředí používá Eclipse. thenewboston postupuje od základů velmi jednoduchým způsobem a nazabíhá do hloubky.

Videa:

- 1 - Installing the JDK (http://www.youtube.com/watch?v=Hl-zzrqQoSE) (zveřejněno 2009-05-02)
- 2 - Running a Java Program (http://www.youtube.com/watch?v=5u8rFbpdvds) (zveřejněno 2009-05-02)
- 3 - Downloading Eclipse (http://www.youtube.com/watch?v=CE8UIbb_4iM) (zveřejněno 2009-05-02)
- 4 - Hello YouTube (http://www.youtube.com/watch?v=SHIT5VkNrCg) (zveřejněno 2009-05-02)
- 5 - Variables (http://www.youtube.com/watch?v=gtQJXzi3Yns) (zveřejněno 2009-05-02)
- 6 - Getting User Input (http://www.youtube.com/watch?v=5DdacOkrTgo) (zveřejněno 2009-05-02)

- 7 - Building a Basic Calculator (http://www.youtube.com/watch?v=ANuuSFY2BbY) (zveřejněno 2009-05-02)
- 8 - Math Operators (http://www.youtube.com/watch?v=8ZaTSedtf9M) (zveřejněno 2009-05-03)
- 9 - Increment Operators (http://www.youtube.com/watch?v=ydcTx6idTs0) (zveřejněno 2009-05-03)
- 10 - If Statement (http://www.youtube.com/watch?v=iMeaovDbgkQ) (zveřejněno 2009-05-03)
- 11 - Logical Operators (http://www.youtube.com/watch?v=PAaqgTr7Cx4) (zveřejněno 2009-05-03)
- 12 - Switch Statement (http://www.youtube.com/watch?v=RVRPmeccFT0) (zveřejněno 2009-05-03)
- 13 - While Loop (http://www.youtube.com/watch?v=8ZuWD2CBjgs) (zveřejněno 2009-05-03)
- 14 - Using Multiple Classes (http://www.youtube.com/watch?v=XqTg2buXS5o) (zveřejněno 2009-05-09)
- 15 - Use Methods with Parameters (http://www.youtube.com/watch?v=7MBgaF8wXls) (zveřejněno 2009-05-09)
- 16 - Many Methods and Instances (http://www.youtube.com/watch?v=9t78g0U8VyQ) (zveřejněno 2009-05-09)
- 17 - Constructors (http://www.youtube.com/watch?v=tPFuVRbUTwA) (zveřejněno 2009-05-09)
- 18 - Nested if Statements (http://www.youtube.com/watch?v=Y4xFGCyt1ww) (zveřejněno 2009-05-10)
- 19 - else if Statement (http://www.youtube.com/watch?v=C0YRYVn_BeI) (zveřejněno 2009-05-10)
- 20 - Conditional Operators (http://www.youtube.com/watch?v=Y6NheSwTsDs) (zveřejněno 2009-05-10)
- 21 - Simple Averaging Program (http://www.youtube.com/watch?v=KXuQQh6AynQ) (zveřejněno 2009-05-10)
- 22 - for Loops (http://www.youtube.com/watch?v=rjkYAs6gAkk) (zveřejněno 2009-05-10)
- 23 - Compound Interest Program (http://www.youtube.com/watch?v=T9TcAm9g0mo) (zveřejněno 2009-05-10)
- 24 - do while Loops (http://www.youtube.com/watch?v=nfr52iR0Pyg) (zveřejněno 2009-05-10)
- 25 - Math Class Methods (http://www.youtube.com/watch?v=JzMdepMLW44) (zveřejněno 2009-05-10)
- 26 - Random Number Generator (http://www.youtube.com/watch?v=AhwIYAXPASw) (zveřejněno 2009-05-10)
- 27 - Introduction to Arrays (http://www.youtube.com/watch?v=L06uGnF4IpY) (zveřejněno 2009-05-10)
- 28 - Creating an Array Table (http://www.youtube.com/watch?v=nTF-RcgsV0E) (zveřejněno 2009-05-10)
- 29 - Summing Elements of Arrays (http://www.youtube.com/watch?v=etyrkipdKvc) (zveřejněno 2009-05-15)
- 30 - Array Elements as Counters (http://www.youtube.com/watch?v=pHxtKDENDdE) (zveřejněno 2009-05-15)
- 31 - Enhanced for Loop (http://www.youtube.com/watch?v=w41D0V-BnKQ) (zveřejněno 2009-05-15)
- 32 - Arrays in Methods (http://www.youtube.com/watch?v=rzXoz2KOP7E) (zveřejněno 2009-05-16)
- 33 - Multidimensional Arrays (http://www.youtube.com/watch?v=ctab5xPv-Vk) (zveřejněno 2009-05-16)
- 34 - Table for Multi Arrays (http://www.youtube.com/watch?v=hbot9MQVHOM) (zveřejněno 2009-05-16)
- 35 - Variable Length Arguments (http://www.youtube.com/watch?v=BFL1oWnEO2k) (zveřejněno 2009-05-16)
- 36 - Time Class (http://www.youtube.com/watch?v=o4Or0PMI_aI) (zveřejněno 2009-08-02)
- 37 - Display Regular time (http://www.youtube.com/watch?v=E0BTAqIltFc) (zveřejněno 2009-08-02)
- 38 - Public, Private and this (http://www.youtube.com/watch?v=csjfLTt6-io) (zveřejněno 2009-08-02)
- 39 - Multiple Constructors (http://www.youtube.com/watch?v=LS7BzkBzn3Y) (zveřejněno 2009-08-02)
- 40 - Set and Get Methods (http://www.youtube.com/watch?v=eqP5X6APc5w) (zveřejněno 2009-08-02)
- 41 - Building Objects for Constructors (http://www.youtube.com/watch?v=MK2SMJZbUmU) (zveřejněno 2009-08-02)
- 42 - toString (http://www.youtube.com/watch?v=l0N6WvIVoUI) (zveřejněno 2009-08-07)
- 43 - Composition (http://www.youtube.com/watch?v=ZBkyPA6NZR8) (zveřejněno 2009-08-07)
- 44 - Enumeration (http://www.youtube.com/watch?v=uFGrL5vyp54) (zveřejněno 2009-08-08)
- 45 - EnumSet range (http://www.youtube.com/watch?v=r-_6fJpC-pk) (zveřejněno 2009-08-08)
- 46 - Static (http://www.youtube.com/watch?v=Mhxp5dZOy78) (zveřejněno 2009-08-08)
- 47 - More on Static (http://www.youtube.com/watch?v=14c1oJjgC8g) (zveřejněno 2009-08-08)

- 48 - final (http://www.youtube.com/watch?v=Suxdg95FV1w) (zveřejněno 2009-08-08)
- 49 - Inheritance (http://www.youtube.com/watch?v=9JpNY-XAseg) (zveřejněno 2009-08-09)
- 50 - Graphical User Interface GUI (http://www.youtube.com/watch?v=jJjg4JweJZU) (zveřejněno 2009-09-16)
- 51 - GUI with JFrame (http://www.youtube.com/watch?v=jUdIAgJ7JKo) (zveřejněno 2009-09-19)
- 52 - Event Handling (http://www.youtube.com/watch?v=3EE7E3bvfe8) (zveřejněno 2009-09-20)
- 53 - ActionListner (http://www.youtube.com/watch?v=qhYook53olE) (zveřejněno 2009-09-20)
- 54 - Event Handler Program (http://www.youtube.com/watch?v=M1_-sigEPtE) (zveřejněno 2009-09-20)
- 55 - Intoduction to Polymorphism (http://www.youtube.com/watch?v=0xw06loTm1k) (zveřejněno 2009-09-23)
- 56 - Polymorphic Arguements (http://www.youtube.com/watch?v=KKbN5pjBZGM) (zveřejněno 2009-09-23)
- 57 - Overriding Rules (http://www.youtube.com/watch?v=zN9pKULyoj4) (zveřejněno 2009-09-24)
- 58 - Abstract and Concrete Classes (http://www.youtube.com/watch?v=TyPNvt6Zg8c) (zveřejněno 2009-09-24)
- 59 - Class to Hold Objects (http://www.youtube.com/watch?v=slY5Ag7IjM0) (zveřejněno 2009-09-27)
- 60 - Array Holding Many Objects (http://www.youtube.com/watch?v=0--h2x6HENA) (zveřejněno 2009-09-27)
- 61 - Simple Polymorphic Program (http://www.youtube.com/watch?v=6d0m_L8_1XU) (zveřejněno 2009-10-02)
- 62 - JButton (http://www.youtube.com/watch?v=6iV-v_m0z0w) (zveřejněno 2009-10-02)
- 63 - JButton Final Program (http://www.youtube.com/watch?v=3RQOikbGGUM) (zveřejněno 2009-10-02)
- 64 - JCheckBox (http://www.youtube.com/watch?v=_UuDuj-RNRg) (zveřejněno 2009-10-03)
- 65 - The Final Check Box Program (http://www.youtube.com/watch?v=Y8zKDsenQFA) (zveřejněno 2009-10-03)
- 66 - JRadioButton (http://www.youtube.com/watch?v=_d4CU9MveLE) (zveřejněno 2009-10-03)
- 67 - JRadioButton Final Program (http://www.youtube.com/watch?v=-ptlsT9KsM8) (zveřejněno 2009-10-03)
- 68 - JComboBox (http://www.youtube.com/watch?v=vd-k2oBMXUI) (zveřejněno 2009-10-03)
- 69 - Drop Down List Program (http://www.youtube.com/watch?v=XS4-5GmRnp8) (zveřejněno 2009-10-03)
- 70 - JList (http://www.youtube.com/watch?v=GBlKa8cNROM) (zveřejněno 2009-10-05)
- 71 - JList Program (http://www.youtube.com/watch?v=aLkkYbHz16E) (zveřejněno 2009-10-05)
- 72 - Multiple Selection List (http://www.youtube.com/watch?v=9z_8yEv7nIc) (zveřejněno 2009-10-05)
- 73 - Moving List Items Program (http://www.youtube.com/watch?v=68X8RUxeXeA) (zveřejněno 2009-10-05)
- 74 - Mouse Events (http://www.youtube.com/watch?v=hsHqhX0s7Rs) (zveřejněno 2009-10-10)
- 75 - MouseListener interface (http://www.youtube.com/watch?v=MpIHF4V3zMc) (zveřejněno 2009-10-10)
- 76 - MouseMotionListener interface (http://www.youtube.com/watch?v=sdUJR_DSyBU) (zveřejněno 2009-10-10)
- 77 - Adapter Classes (http://www.youtube.com/watch?v=UuKNGMCfSkQ) (zveřejněno 2009-10-11)
- 78 - File Class (http://www.youtube.com/watch?v=7fC9nL3_AQQ) (zveřejněno 2009-10-23)
- 79 - Creating Files (http://www.youtube.com/watch?v=G0DfmD0KKyc) (zveřejněno 2009-10-23)
- 80 - Writing to Files (http://www.youtube.com/watch?v=Bws9aQuAcdg) (zveřejněno 2009-10-23)
- 81 - Reading from Files (http://www.youtube.com/watch?v=3RNYUKxAgmw) (zveřejněno 2009-10-23)
- 82 - Exception Handling (http://www.youtube.com/watch?v=K_-3OLkXkzY) (zveřejněno 2009-10-25)
- 83 - FlowLayout (http://www.youtube.com/watch?v=DFQzFJqOSbA) (zveřejněno 2009-10-30)
- 84 - Drawing Graphics (http://www.youtube.com/watch?v=2l5-5PMUc5Y) (zveřejněno 2009-10-30)
- 85 - JColorChooser (http://www.youtube.com/watch?v=052U-bWEXrk) (zveřejněno 2009-10-30)
- 86 - Drawing More Stuff (http://www.youtube.com/watch?v=OWOeE90ET6w) (zveřejněno 2009-10-30)

74.2.2. Knihy, články a jiné materiály

Effective Java Programming by Joshua Bloch

- Effective Java Programming with Joshua Bloch (http://www.youtube.com/watch?v=ZOwHiGCzZjo) na YouTube

74.3. Eclipse

Odkazy:

- eclipse refuses to start (http://forums.debian.net/viewtopic.php?f=6&t=36174)
- Eclipse integrated browser on Lenny (http://forums.debian.net/viewtopic.php?f=6&t=33452&start=0)

Eclipse mi v Lenny nefungovalo. Padalo na ošklivou chybu. Hledal jsem na internetu a nakonec narazil na fórum (http://forums.debian.net) a diskusi Eclipse integrated browser on Lenny (http://forums.debian.net/viewtopic.php?f=6&t=33452&start=0). S těmito informacemi jsem použil následující postup. Nejdříve jsem nainstaloval potřebné balíčky.

```
# aptitude install icedove openjdk-6-jdk eclipse
```

Poté jsem si nakopíroval do vlastního bin adresáře spouštěcí skript eclipse.

```
$ cp /usr/bin/eclipse ~/bin/
```

V tomto skriptu provedl úpravu která je vidět na diffu.

```
$ diff -U3 /usr/bin/eclipse /home/radek/bin/eclipse
--- /usr/bin/eclipse 2009-09-10 18:39:11.000000000 +0200
+++ /home/radek/bin/eclipse 2009-09-10 18:51:27.779433488 +0200
@@ -142,6 +142,8 @@
     export MOZILLA_FIVE_HOME=/usr/lib/mozilla-firefox
 elif [ -e /usr/lib/mozilla/libgtkembedmoz.so ]; then
     export MOZILLA_FIVE_HOME=/usr/lib/mozilla
+elif [ -e /usr/lib/icedove/libgtkembedmoz.so ]; then
+    export MOZILLA_FIVE_HOME=/usr/lib/icedove
 else
     $DIALOGW \
         --title="Integrated browser support not working" \
```

Eclipse startuje.

74.3.1. Instalace aktuální verze

Potřebujeme-li aktuálnější verzi Eclipse, než je v repositářích Debianu, můžeme použít následující postup.

Podíváme se na web Eclipse.org (http://www.eclipse.org) a najdeme příslušný soubor ke stažení. Já jsem tam našel Eclipse IDE for Java EE Developers (188 MB)

(http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/galileo/SR1/eclipse-jee-galileo-SR1-linux-gtk-x86_64.tar.gz). Stažený balíček otevřeme, a obsah adresáře `eclipse` nakopírujeme například do `~/lib/java/exlipse-3.5.1`. Nyní zbývá jen vytvořit položku v menu. Tu můžeme vytvořit tak, že v adresáři `~/.local/share/applications` vytvoříme soubor `eclipse-3.5.1.desktop` s následujícím obsahem.

```
[Desktop Entry]
Name=Eclipse 3.5.1 Galileo
Exec=/home/radek/lib/java/eclipse-3.5.1/eclipse
Icon=/home/radek/lib/java/eclipse-3.5.1/plugins/org.eclipse.platform_3.3.201.v200909170800/ecl
Categories=Application;Development;Java;IDE
Encoding=UTF-8
Version=1.0
Type=Application
Terminal=false
StartupNotify=trueq
```

74.4. Prostředí

74.5. Jazyk Java

To be done:

74.5.1. Proměnné a základní datové typy

int

Datový typ reprezentuje celá čísla v rozsahu od -2^{31} do $2^{31}-1$. V paměti zaujímá 32 bitů (4 bajty). Příklady hodnot: `5`, `18`, `-2`.

double

Datový typ reprezentující reálná čísla v dvojnásobném rozsahu. Reálná čísla jsou reprezentována typem float. Příklady hodnot: `3.14`.

byte

short

char

long

float

boolean

Tabulka 74-1. Primitivní typy v Javě

byte	-128	127	8 bits
short			16 bits
int			32 bits
long			64 bits
char	’\u0000’ (=0)	’\uFFFF’ (=65535)	16 bits

float	IEEE 754	32 bitů
double	IEEE 754	64 bitů
Boolean Type		

74.5.2. Operace se základními typy

Tabulka 74-2.

priorita	arita	operátor	symbol	asociativita
		priorita	( )	
	unary	postfix	*expr*++, *expr*--	N/A
	unary	prefix	++*expr*, --*expr*	right to left
	unary		-*expr*, +*expr*, ~*expr*, !*expr*	right to left
	binary	multiplicative	*, /, %	left to right
	binary	additive	+, -	left to right
	binary	shift	<<, >>, >>>	left to right
	binary	relational	<, <=, >=, >, instanceof	left to right
	binary	equality	==, !=	left to right
	binary	bitwise/logical AND	&	left to right
	binary	bitwise/logical XOR	^	left to right
	binary	bitwise/logical OR	|	left to right
	binary	logical AND	&&	left to right
	binary	logical OR	||	left to right

priorita	arita	operátor	symbol	asociativita
	ternary		? :	right to left
		assignment	=, +=, -=, *=, /=, %=, ^=, |=, <<=, >>=, >>>=, &=	right to left

```
i = -j;
```

74.5.3. Programové konstrukce pro řízení toku programu

74.5.3.1. Větvení

```
int test = 6;
if ( test != 9) {
    System.out.println("yes");
} else {
    System.out.println("this is else");
}
```

```
int age;
age = 3;

switch (age) {
case 1:
    System.out.println("You can crawl");
    break;
case 2:
    System.out.println("You can talk");
    break;
case 3:
    System.out.println("You can get in trouble");
    break;
default:
    System.out.println("I dont know how old you are");
    break;
}
```

74.5.3.2. Cykly

```
int counter = 0;
while (counter < 10) {
    System.out.println(counter);
    counter++;
}
```

```
import java.util.Scanner;

public class Main {
 public static void main(String[] args) {
  Scanner input = new Scanner(System.in);
  int total = 0, grade, counter = 0;
```

```
  double average;

  while (counter < 10) {
   grade = input.nextInt();
   total = total + grade;
   counter++;
  }
  average = total / 10.0;
  System.out.println("Your average is " + average);
 }
}
```

For cyklus

```
for(int counter=1; counter <=10; counter+=3) {
    System.out.println(counter);
}
```

74.5.3.2.1. while

while (*condition*)
  statement

74.5.3.2.2. do loop

do {
  body
}while(*condition*)

74.5.4. Třídy

Odkazy:

-
- Java Tutorial 3.2 Classes Part 2/12 - classes (http://www.youtube.com/watch?v=TNPcfVGVjJ8)
- Java Tutorial 3.3 Classes Part 3/12 - classes (http://www.youtube.com/watch?v=J_PCMJjBUjY)
-
-

```
class BankAccount {
        /*
         * Fields, atributes
         */
        long    accountNumber;
        double  balance;
        String  customerName;

        /*
```

```
         * Constructor
         */
        BankAccount(long number, double openingBal, String customer) {
                // Code to create a new BankAccount object
                accountNumber = number;
                balance = openingBal;
                customerName = customer;
        }

        /*
         * Methods
         */
        void deposit(double amount) {
                // code to deposit amount
                balance += amount;
        }

        void withdraw(double amount) {
                // code to withdraw amount
                balance -= amount;
        }

        // any additional methods, constructors and/or fields
}
```

Použití objektů třídy BankAccount. Nejdříve vytvoření objektů.

```
BankAccount  account1 = new BankAccount(123456, 1E6, "J DOe");
BankAccount  account2 = new BankAccount(123457, 112E6, "D Beckham");

double difference = account2.balance - account1.balance;

System.out.println(account2.customerName + " has " + difference + " more than " + account1.cus

account2.withdraw(difference);
account1.deposit(difference);

account1 = account2;
```

74.5.4.1. Dědičnost *Inheritance*

Odkazy:

- Java Tutorial 3.4 Classes Part 4/12 - inheritance (http://www.youtube.com/watch?v=2-I70Q6azmU)
-
-

class MyClass **extends** OtherClass {

Třída, `MyClass`, dědí pouze z jedné třídy `OtherClass`. O třídě `MyClass` říkáme že je podtřídou třídy `OtherClass`. Třída může mít libovolné množství podtříd.

```
class Person {
        String name;
        String address;
        Date   dateOfBirth;
```

```
        // methods for class person
}

class Employee exptends Person {
        float   salary;
        Employee immediateManager;
        // methods for employee class
}

class Manager extends Employee {
        String departement;
        // methods for manager class
}
```

74.5.4.2. Atributy a metody třídy

Klíčovým slovem *static* před deklaraci proměnné nebo metody ve třídě změníme vlastnost této. Proměnná se stává proměnnou třídy a nikoliv objektu (instance) a je společná všem instancím.

74.5.5. Viditelnost a rozsahy platnosti

private / public

74.5.6. Statické proměnné

static

74.5.7. Packages

Odkazy:

- Java Tutorial 4.1 Packages Part 1/12 - interfaces (http://www.youtube.com/watch?v=LQ8meqMr9LU)
-
- Java Tutorial 4.3 Packages Part 3/12 - package names (http://www.youtube.com/watch?v=y0U_EkLNH4c)
-
-
-

package *name*;

```
package com.acmewidgets.client.security;
```

Konvence pro tvorbu jména balíčku jsou:

- Použijte doménové jméno vaší společnosti a napiště jej obráceně. Máme li tedy jméno `example.com` napíšeme `com.example`.
- Doplňte vlastním názvem balíčku podle vlastní hierarchie.

74.5.7.1. Interface

```
interface Drawable {
        void draw(float x, float y);
        void erase();
        void move(float x, float y);
}

class Square implements Drawable {
}
class Circle implemets Drawable {
}
```

74.5.7.2. Viditelnost tříd

Odkazy:

- Java Tutorial 4.4 Packages Part 4/12 - access qualifiers (http://www.youtube.com/watch?v=NU8pZ9oSNPU)

Třídy jsou viditelné jen v rámci balíčku. Pokud třídu chceme zpřístupnit mimo balíček, použijeme klíčové slovo `public`.

```
package com.example.mujbalicek;
    class NonPublic { /* */ } // Může být použita jen v rámci balíčku
    public class Test { /* */ } // Je publikována pro veřejné použití ostatními balíčky
```

74.5.8. Pole (*Array*)

Deklarace.

```
int[] ia; //array of integers
int ia[]; // stejné jako předchozí řádek, nedoporučuje se
```

Deklarace vytváří proměnnou ale nikoliv objekt, v našem případě pole. Pokud chceme pole vytvořit můžeme v deklaraci použít inicializaci.

```
int[] ia = new int[5];
MyType[][] mt = new MyType[4][16];
```

Přístup k prvkům pole.

```
int[] ary = new int[4];
ary[0] = 345;
i = ary[3];
```

Zápis literálu pole.

```
int[] ar = new int[6];
ar = {0, 10, 20, 30, 40, 50};
```

Inicializace pole hodnotami při deklaraci. Tedy v deklaraci pole vytvoříme a inicializujeme.

```
int a[] = {1, 2, 3, 5, 7, 11, 13, 17, 19};
```

74.5.9. Řetězce (*String*)

Odkazy:
- Lecture 9 | Programming Methodology (Stanford) (http://www.youtube.com/watch?v=iYtri45lhtc)

74.5.10. Výjimky

```
try {
   // code here
} catch(exceptionType1 parameter1) {
   // code here
}
...
} finally {
   // code here
}
```

```
void myMethod(myType T) throws IOException {
    // code
}
```

74.5.11. assert

Používá se s výhodou při ladění programů. Jako `Expression1` použijeme nějakou invariantu.

```
assert Expression1;
assert Expression1: Expression2;
```

74.5.12. synchronized

```
synchronized (Expression) {
   // code
}
```

```
public static synchronized int testExample() { /* code */ }
```

74.5.13. Abstraktní třídy

Abstraktní třída je třída jejíž objekty nemohou být vytvořeny. Je to vlastně jakási šablona, kterou musíme vyplnit konkrétní implementací.

V následující ukázce použití abstraktní třídy programujeme třídu `Shape` jako abstraktní. To říká že nemůžeme vytvořit žádný objekt této třídy. Vytvoření (instanciace) objektu třídy `Shape` je nesmyslné. Poté naprogramujeme třídu `Circle` jako podtřídu třídy `Shape`. Objekty třídy `Circle` již mohou být vytvářeny.

```
abstract class Shape {
        private int locationX, locationY;
```

```
        private Color color;
        private int lineThickness;

        setLocation(int x, int y) {locationX=x; locationY=y;}
        setColor(Color col) {color = col;}
 abstract void drawShape();  // Abstraktní metoda.
        // abstraktní metoda MUSÍ být v podtřídě implementována.
        /* ostatní metody */
}

class Circle extends Shape {
        int radius;
        Circle(int x, int y, int rad) {
                radius = rad;
                setLocation(x,y);
        }

        drawShape() {
                // code here
        }
}


public abstract class Number implementrs java.io.Serializable {

    /* Returns the value as a various primitive types (this may involve rounding or truncation

    public abstract int intValue();
    public abstract long longValue();
    public abstract float floatValue();
    public abstract double doubleValue();

    public byte byteValue() {
        return (byte)intValue();
    }

    public short shortValue() {
        return (short)intValue();
    }

    /** use serialVersionUID from JDK 1.0.2 for interoperability */
    private static final long serialVersionUID = -8742448824652078965L
}


interface I1 {
    void i1M1();
    int  i1m2(I1 p1, float p2);
}

interface I2 {
        abstract int i2M2();
}

interface I3 {
        I3 i3M1(I2 p1) {
}
```

```
interface I4 extends I3,I1 {
        void i4M1();
}

abstract class C1 implements I1, I2 {
        void i1M1() {/* code */}
        abstract void c1M2();
}

abstract class C2 extends C1 implements I2 {
}

class C3 extends C2 implementrs I4 {
        // code
}

// C3 requires code for the following methods c1M2, i1M2, i2M2, i4M1 and i3M1.
```

74.6. Prostředí

* *Nejsem si jist názvem této kapitoly.*

Odkazy:

- Robocode (http://robocode.sourceforge.net/) na SourceForge
- Robocode (http://robocoderepository.com/HomePage.jsp)
- Greenfoot (http://www.greenfoot.org/index.html)

74.6.1. Robocode

Odkazy:

- RoboWiki (http://robowiki.net/wiki/Main_Page)

74.7. Javadoc

Nástroj pro vytváření dokumentace ke kódu psanému v Javě. Tento nástroj prochází kód a ze správně formátovaných a označených komentářů a s pomocí klíčových slov vytváří dokumentaci.

74.7.1. Dokumentování getters/setters

```
/** The main register */
private int register;

/** Setter for {@link #register} */
public void setRegister(int value);

/** Getter for {@link #register} */
public in getRegister();
```

74.8. Krátké statě

Krátké texty věnované různým aspektům programování v Javě.

74.8.1. Použití tříd jako jsou v C/C++ použity struktury

Odkazy:

- Substitutes for Missing C Constructs (http://java.sun.com/developer/Books/shiftintojava/page1.html) z knihy Effective Java Programming

```
// Degenerovaná třída.  Nesmí být nikdy veřejná!
class Point {
    public float x;
    public float y;
}
```

Tento přístup může být některým programátorům proti srsti. Chtěli by třídu deklarovat s privátními `x` a `y`. s použitím přístupových (getX/setX) funkcí. Proto je třeba mít na paměti že degenorovaná třída slouží jako náhrada za strukturu.

74.8.2. Nahrazení unionů hierarchií tříd

74.9. Nezpracované informace

74.9.1. Rozšíření pole

```
// a je pole
    int length = a.length;
    int[] n_a = new int[length+1];
    for (int n=0; n<length; n++) {
        n_a[n] = a[n];
    }
    n_a[length] = 13; // Nová hodnota přidaná na konec pole
    a = n_a;
```

Jépe použít knohovnu java.util.ArrayList

```
import java.util.ArrayList;
public class Main {
    public static ArrayList a;

    public static void main(String[] args) {
        a = new ArrayList();
        a.add(66);
        System.out.println(Integer.toString(a.size()));
    }
}
```

Kapitola 75. Programování v assembleru

* *Toto jsou jen neutříděné poznámky, nikoli povídání o programování. Jediné co by zde snad mohlo mít hodnotu jsou odkazy.*

Odkazy:

- Using as, the Gnu Assembler (http://linux.web.cern.ch/linux/scientific4/docs/rhel-as-en-4/index.html)
- Linux Assembly HOWTO (http://www.faqs.org/docs/Linux-HOWTO/Assembly-HOWTO.html) by Konstantin Boldyshev, 2002
- linuxassembly.org (http://asm.sourceforge.net/)
-
-
-

Programy:

- readelf
-
-
-
-

Program v assembleru má strukturu. Jednotlivé části kódu a deklarací proměnných a konstant se přařazují k sekcím.

.data

: Inicializovaná data. Data která mají defaultní obsah.

.bss

: Neinicializovaná data.

.text

: Kód (text) programu.

75.1. Jednoduchý příklad

*

Následující program napsaný v assembleru pro architekturu amd64 (x86-64) nedělá vůbec nic. Je to jen pokus o nejednodušší kód do kterého pak mohu doplňovat další instrukce.

```
# File: examples/as/hello.s
 .section .data
 .section .text
 .global _start
_start: # Entry point

 # _exit()
 xorq %rax, %rax  # %rax = 0
 incq %rax  # %rax = 1
 xorq %rbx, %rbx  # %rbx = 0
 int $0x80
```

```
$ as -o hello.o hello.s
$ ld -o hello -O0 hello.o
$ ./hello
```

75.2. AMD64

* *Zvláštnosti architektury amd64 (x86-64)*

Odkazy:

- Basic differences between x86 Assembly and X86-64 Assembly (http://www.milw0rm.com/papers/110)
-
- x86_64 assembly help (http://www.linuxquestions.org/questions/programming-9/x86_64-assembly-help-524748/)
-

Small code model

Code from 0x0 to 0x7effffff. *Fastest code model.*

Kernel code model

Code from 0xffffffff80000000 to 0xffffffffff000000.

Medium code model

Stejný jako Small code model. Možnost navíc large data sections. (.ldata, .lrodata, .lbss). Překladač musí používat instrukce **movabs** pro přístup large statickým datům a pro nahrávání adres do registruů.

Large code model

small position independent code model (PIC)

Medium position independent code model (PIC)

Large position independent code model (PIC)

Příklad 75-1. Použití 64-bit systémových volání

```
# File: example/as/amd64/hello64.s

 .section .data
msg: .ascii "Hello, world!\n"
len: .long . - msg

 .section .text
 .global _start
_start: # Program starting point.  Linker looks for _start symbol.

 pushq $14; pop %rdx # %rdx=14
 pushq $msg; pop %rsi # %rsi=msg
```

```
 pushq $1;  pop %rdi # %rdi=1 (STDOUT)
 pushq $1;  pop %rax # write(rdi,rsi,rdx)
 syscall

 pushq $2; pop %rdi # retval=1
 pushq $0x3c; pop %rax # exit()
 syscall
```

Příklad 75-2. Použití 32-bit systémových volání

```
# File: examples/as/amd64/hello.s
 # Použitá volání
 .equ SYS_exit, 1
 .equ SYS_write, 4

 # Standardní proudy
 .equ STDIN, 0
 .equ STDOUT, 1
 .equ STDERR, 2

 .section .data
msg: .ascii "Hello, world!\n"
len: .long . - msg

 .section .text
 .global _start
_start: # Program starting point.  Linker looks for _start symbol.

 movl $SYS_write, %eax # sys_write(ebx=fd, ecx=ptr, edx=len)
 movl $STDOUT, %ebx # stdout
 leal msg, %ecx # Message start point
 movl len, %edx # Message lenght
 int $0x80  # call kernel

 movl $SYS_exit, %eax # sys_exit()
 movl $1, %ebx # return value (only 8 bits)
 int $0x80  # call kernel
```

75.3. LP64

Podle této poznámky z www.treblig.org (http://www.treblig.org/articles/64bits.html) je organizace LP64 mixem 64 a 32 bitů. To nás může velmi mást.

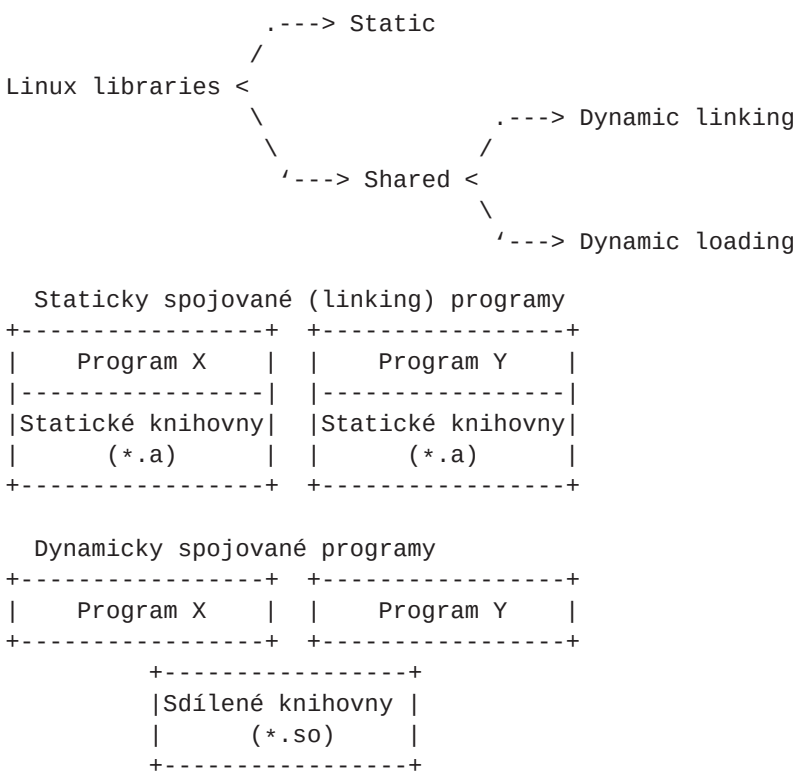
> **Poznámka:** 64 bit Linux machine use an organisation called LP64 where 'long's and pointers are 64 bit in length but everything else is 32 bit; long long's are also 64 bit. 32 bit Linux systems have both long's and pointers as 32 bit and have the 'long long' type as 64 bit.

75.4. Různé poznámky

*

75.4.1. Knihovny

*



V případě statického spojování programu s knihovnami se spojování provádí v době překladu programu. Knihovna je součástí binárního souboru programu.

V případě dynamického spojování není knihovna součástí programu. V programu je jen seznam symbolů (proměnných a funkcí) které jsou definovány v knihovně. Při zavádění programu do paměti počítače zavaděčem tento vyhledá knihovnu a spojí symboly v programu s funkcemi a proměnnými v knihovně. Pokud je knihovna již v paměti, nemusí se zavádět znova. Všechny běžící programy tak sdílí jeden kód knihovny v paměti počítače.

75.4.2. Volání systému (Linuxového jádra)

*

Systém se volá instrukcí softwarového přerušení **int $0x80**. V registru `%eax` je číslo služby kterou voláme a v registrech `%ebx`, `%ecx`, `%edx`, `%esi`, `%edi` a `%ebp` jsou parametry volání. Tímto způsobem je možno předat až 6 parametrů. Pokud předáváme parametrů více, předávají se pomocí ukazatele na blok paměti. Tento ukazatel uložíme do registru `%ebx`. Návratová hodnota volání je obvykle vrácena v registru `%eax`.

```
movl    $1, %eax        # Číslo volání/služby
movl    $0, %ebx
int     $0x80
```

Čísla jedotlivých volání najdeme v hlavičkovém souboru `/usr/include/asm/unistd.h`. V případě architektury intel/amd jsou to hlavičkové soubory `/usr/include/asm/unistd_32.h` a `/usr/include/asm/unistd_64.h`.

Tabulka 75-1. Tabulka vybraných systémových volání jádra

%eax	název		parametry
0	sys_restart_syscall		
1	sys_exit		ebx=int
2	sys_fork	`arch/x86/kernel/process_32.c`	ebx=struct pt_regs
3	sys_read		
4	sys_write		
5	sys_open		
6	sys_close		
7	sys_waitpid		
8	sys_creat		
9	sys_link		
10	sys_unlink		
11	sys_execve		
12	sys_chdir		
13	sys_time		
14	sys_mknod		
15	sys_chmod		
16	sys_lchown		
17	sys_break		
18	sys_oldstat		
162	sys_nanosleep		
180	pread64		
181	pwrite64		
0	sys_		
0	sys_		
321	sys_signalfd		
322	sys_timerfd_create		
323	sys_eventfd		
324	sys_fallocate		
325	sys_timerfd_settime		
326	sys_timerfd_gettime		

write

Jméno

`write` —

*

Ukázky použití

Příklad 75-1. Použití 32-bitového volání write

```
movl $4, %eax # sys_write(ebx=fd, ecx=ptr, edx=len)
movl $1, %ebx # stdout
leal msg, %ecx # Message start point
movl len, %edx # Message lenght
int $0x80  # call kernel
```

75.4.3. 64-bitová volání jádra architektury amd64 (x86-64)

*

Odkazy:

- Basic differences between x86 Assembly and X86-64 Assembly (http://www.milw0rm.com/papers/110)
- Gentle Introduction to x86-64 Assembly (http://www.x86-64.org/documentation/assembly)
-
-

Tabulka 75-2.

reg.	použití		
%rax	dočasný registr	ne	

```
endMeasure:
mov rsi,[string]
mov rax,1
syscall

mov rdi,1
mov rax,3ch
syscall
```

Compile assembly code with:

```
$ as --64 code.s -o code.o
$ ld -m elf_x86_64 code.o -o code
```

Compile C code with:

```
$ gcc -m64 code.c -o code
```

Kapitola 76. Ladění programu

*

Odkazy:

- Debugging on Linux (http://www.treblig.org/articles/debugtalk/index.html)
-
-

Pro ladění slouží program **gdb**. Ten má textové rozhraní. Můžeme ale také použít některou z grafických nadstaveb nad **gdb** jako jsou například **ddd** a **insight**.

- **ddd** — Data Dispaly Debugger
- **insight** — Graphical debugger based on GDB

Rovněž editor Emacs má rozhraní pro gdb.

Pokud chceme při ladění vidět symboly, tedy názvy funkcí a proměnných jak jsou v programu, je třeba při překladu použít přepínač `-g`.

Další nástroje které můžeme použít jsou:

- Ltrace
- Strace

Kapitola 77. Různé

77.1. Pojmenované roury / named pipes

Pojmenované roury vytváříme příkazem mknod nebo mkfifo:

```
$ mknod moje-roura p
$ mkfifo dalsi-roura
```

V programu můžeme využít volání

int mkfifo(const char *path, mode_t mode)

```
mknod npipe p
echo "neco a jeste neco jineho" >npipe &
cat npipe
rm npipe
```

IX. C

FIXME:napsat.

Kapitola 78. Volání jádra operačního systému

* ***To be done:***

Kapitola 79. libc

* ***To be done:***

  Odkazy a zdroje:

  - The GNU C Library (http://www.gnu.org/software/libc/manual/html_node/index.html)

79.1. Ošetření chyb

Většina funkcí knihoven má vyhrazenu zvláštní hodnotu pro indikaci chyby.

Funkce které vracejí hodnotu typu int většinou indikují výskyt chyby návratovou hodnotou `-1`. Funkce vracející ukazatel pak používají obvykle k oznámení chyby návratovou hodnotu `NULL`.

Konkrétní chyba je pak určena proměnnou `errno` deklarovanou v hlavičkovém souboru `errno.h`. V tomto souboru je obvykle `errno` deklarována jako makro. Obsah této proměnné je definován jen v případě chyby. Dále jsou v hlavičkovém souboru deklarovány konstanty pro jednotlivé chyby.

K dispozici jsou dále funkce které vracejí textový popis chyby. Jsou to.

```
#include <stdio.h>
void perror(const char *s);
char *strerror(int errnum);
int strerror_r(int errnum, char *buf, size_t n);
```

Funkce `perror()` vypíše chybové hlášení a popis chyby na standardní chybový výstup `stderr`.

* ***To be done:***

79.2. Práce se soubory

```
#include <sys/tapes.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(const char* pathname, int flags);
int open(const char* pathname, int flags, mode_t mode);

#include <sys/types.h>
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, cosnt void *buf, size_t count);
off_t lseek(int fildes, off_t offset, int whence);

SEEK_SET
SEEK_CUR
SEEK_END

#include <unistd.h>
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

```
STDIN_FILENO
STDOUT_FILENO
STDERR_FILENO
```

* ***To be done:***

79.3. Síťová komunikace

* ***To be done:***

#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol)

79.3.1. Jednoduchý TCP server

Příklad 79-1.

```
/*
 *
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>

int main(argc, argv)
{
 int s, fd, len;
 struct sockaddr_in my_addr;
 struct sockaddr_in remote_addr;
 int sin_size;
 char buf[BUFSIZ];

 memset(&my_addr, 0, sizeof(my_addr));
 my_addr.sin_family = AF_INET;
 my_addr.sin_addr.s_addr = INADDR_ANY;
 my_addr.sin_port = htons(8000);
 if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
  perror("socket");
  return 1;
 }
 if (bind(s, (struct sockaddr *) &my_addr, sizeof(struct sockaddr)) < 0) {
  perror("bind");
  return 1;
 }
 listen(s, 5);
 sin_size = sizeof(struct sockaddr_in);
 if ((fd = accept(s, (struct sockaddr *) &remote_addr, &sin_size)) < 0) {
```

```
  perror("accept");
  return 1;
 }
 printf("accepted client %s\n", inet_ntoa(remote_addr.sin_addr));
 len = send(fd, "Wellcome to my server\n", 21, 0);
 while ((len = recv(fd, buf, BUFSIZ, 0)) > 0) {
  buf[len] = '\0';
  printf("%s\n", buf);
  if (send(fd, buf, len, 0) < 0) {
   perror("write");
   return 1;
  }
 }
 close(fd);
 close(s);
 return 0;
}
```

Kapitola 80. glib

Odkazy:

- GLib Reference Manual (http://library.gnome.org/devel/glib/stable/index.html)

80.1. Práce s řetězci

```
#include <...>
void g_strstrip(string);


gchar* g_strchomp(gchar *string);


gchar* g_strstrip(gchar *string);
```

Kapitola 81. Šifrování a hesla

Věci kolem šifrování a hesel.

* ***To be done:***

  void apr_sha1_base64(const char *clear, int len, char *out);

81.1. Hesla

* ***To be done:***

  - crypt — One-way string encription (hashing) (http://cz.php.net/crypt)

  - crypt: `rqXexS6ZhobKA`
  - md5: `$1$r31.....$HqJZimcKQFAMYayBlzkrA/`
  - md5 apache: `$apr1$qHDFfhPC$nITSVHgYbDAK1Y0acGRnY0`
  -

  - řetězec nezačínající znakem $ v celkové délce ... znaků je standard DES-based encryption with two character salt.
  - `$1$` — MD5 s 12 znakovým salt (`$1$aPBvu2y.$213YVEs8/5m.jMCXSScly/`)
  - `$2$` nebo `$2a$` — BLOWFISH s 16 znakovým salt

  def md5crypt(password, salt, magic='1')

```
import md5

def md5crypt(password, salt, magic='$1$'):
    m = md5.new()
    m.update(password + magic + salt)

    mixin = md5.md5(password + salt + password).digest()
    ...
```

Kapitola 82. Knihovna funkcí

* ***To be done:***

82.1. sleep

Odkazy:

- Sleeping (http://www.delorie.com/gnu/docs/glibc/libc_445.html) in The GNU C Library
- Sleeping (http://www.gnu.org/software/libc/manual/html_node/Sleeping.html)

```
#include <unistd.h>
unsigned int sleep( unsigned int seconds );
```

Funkci se předává jako jediný parametr doba v sekundách o kterou chceme pozdržet vykonávání programu, program uspat. Jako návratovou hodnotu dostaneme 0, je-li vše v pořádku, nebo nenulové číslo které nám sděluje, kolik sekund do námy požadované doby zbývá. To v případě, kdy je program ze spánku probuzen předčasně jinou událostí.

Musíme tedy počítat s tím, že se program „probudí“ dříve něž jsme požadovali. Pokud chceme lepší kód, můžeme použít následující kontstrukci která opakovaně požaduje `sleep` dokud není vyčerpán celkový požadovaný čas. I přesto ale musíme mít na paměti že celkový "quote" prospaný čas nemusí přesně sedět. Bude to kvůli kódu navíc a kvůli nepřesnostem kdy žádáme o spánek v celých sekundách ale probuzeni můžeme být v zlomcích sekund.

```
for (t = tot_time_to_sleep; (tleft = sleep(t)) > 0 && errno == EINTR; t = left);
```

Následující kód nebyl odzkoušen.

```
for (n=10; n > 0 && errno == EINTR; n=sleep(n));
```

funkce sleep

Kapitola 83. Knihovny funkcí

UNIX a nejen UNIX je naprogramován z podstatné části v programovacím jazyce C. Myslím tím jak jeho jádro, tak většinu programového vybavení. S tím souvisí i rozsáhlé knihovny funkcí. V této části uvedu jen ty funkce z těch knihoven, o něž jsem se z nějakého důvodu zajímal, nebo jsem je použil.

V holé instalaci chybí manuálové stránky pro funkce ze standardních knihoven. Tyto se nacházejí v balíčku `manpages-dev`, nebo balíčcích `manpages-`*country-code*. Seznam balíčků z jejich popisem získáme jednoduchým dotazem na balíčkovací systém, například přes aptitude

* *Seznam balíčků dostupných pro mou instalaci etch.*

```
$ aptitude search manpages
p   asr-manpages                          - alt.sysadmin.recovery manual pages
p   erlang-manpages                       - Erlang man pages
p   freebsd-manpages                      - Manual pages for a GNU/kFreeBSD system
p   funny-manpages                        - more funny manpages
p   gmt-manpages                          - Manpages for the Generic Mapping Tools
i   manpages                              - Manual pages about using a GNU/Linux system
p   manpages-de                           - German manpages
p   manpages-de-dev                       - German development manpages
i   manpages-dev                          - Manual pages about using GNU/Linux for development
p   manpages-es                           - Spanish man pages
p   manpages-es-extra                     - Spanish extra manpages
p   manpages-fi                           - Finnish man pages
p   manpages-fr                           - French version of the manual pages about using GNU/
p   manpages-fr-dev                       - French version of the development manual pages
p   manpages-fr-extra                     - French version of the manual pages
p   manpages-hu                           - Hungarian manpages
p   manpages-it                           - Contains a collection of Italian man pages
p   manpages-ja                           - Japanese version of the manual pages (for users)
p   manpages-ja-dev                       - Japanese version of the manual pages (for developer
p   manpages-ko                           - Korean version of the manual pages
p   manpages-nl                           - Dutch manpages
p   manpages-pl                           - Polish man pages
p   manpages-pl-dev                       - Polish man pages for developers
p   manpages-posix                        - Manual pages about using POSIX system
p   manpages-posix-dev                    - Manual pages about using a POSIX system for develop
p   manpages-pt                           - Portuguese Versions of the Manual Pages
p   manpages-pt-dev                       - Portuguese Versions of the Manual Pages
p   manpages-ru                           - Russian translations of Linux manpages
p   manpages-tr                           - Turkish version of the manual pages
p   manpages-zh                           - Chinese manual pages
p   xmanpages-ja                          - Japanese manual pages for X

# aptitude install manpages-dev
```

83.1. String

Funkce pro práci s řetězci znaků.

isalnum

Jméno

`isalnum, isalpha, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isxdigit` — Funkce/Makra pro zjišťování informací o znaku, tedy informace o množině znaků do které patří či nepatří.

Přehled

```
#include <ctype.h>
int isalnum(int ch);
int isalpha(int ch);
int isascii(int ch);
int blank(int ch);
int iscntrl(int ch);
int isdigit(int ch);
int isgraph(int ch);
int islower(int ch);
int isprint(int ch);
int ispunct(int ch);
int isspace(int ch);
int isupper(int ch);
int isxdigit(int ch);
```

xxxxx

Popis

Výše uvedené funkce jsou obvykle realizovány jako makra a zjišťují informace o znaku.

- `isalpha` — je znak písmeno (A-Z, a-z)
- `isdigit` — je znak číslice (0-9)
- `isalnum` — je znak písmeno nebo číslice (isalpha || isdigit
-
-
-

Odkazy:

- isprint(3) (https://kerneltrap.org/man/linux/man3/isprint.3) on kernel trap
- Standard C String & Character (http://www-control.eng.cam.ac.uk/~pcr20/www.cppreference.com/stdstring_details.html)
- Technical Reference: Base Operating System and Extensions, Volume 1 (http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ibm.aix.basetechref/doc/basetrf1/ctype.htm)

- CTYPE(3C) STANDARD C LIBRARY (http://www.cptec.inpe.br/sx4/sx4man2/g1ab02e/ctype.3c.html)
- Linux / Unix Command: isprint (http://linux.about.com/library/cmd/blcmdl3_isprint.htm)
-
-

index

Jméno

`index, rindex` — Vyhledání znaku v řetězci znaků.

Přehled

```
#include <strings.h>
char *index(const char *s, int c);
char *rindex(const char *s, int c);
```

Popis

Funkce `index()` vrací ukazatel na první výskyt znaku *c* v řetězci znaků *s*. Funkce `rindex()` funguje obdobně ale vrací ukazatel na poslední výskyt znaku *c* v řetězci *s*.

Návratová hodnota

Obě funkce varcí ukazatel na nalezený znak nebo NULL pokud tento nebyl nalezen.

Odkazy:

-
-

strchr

Jméno

`strchr, strrchr, strchrnul` — Vyhledání znaku v řetězci.

Přehled

```
#include <string.h>
char *strchr(const char *s, int c);
char *strrchr(const char *s, int c);
#define _GNU_SOURCE
#include <string.h>
char *strchrnul(const char *s, int c);
```

Popis

Funkce `strchr()` vrací ukazatel na první výskyt znaku *c* v řetězci *s*. Funkce `strrchr` pak vrací ukazatel na poslední výskyt znaku *c* v řetězci *s*.

Odkazy:

-
-

strtok

Jméno

`strtok` — Vyhledání tokenů v řetězci.

Přehled

```
#include <string.h>
char *strtok(char str, const char *delim);
char *strtok_r(char str, const char *delim, char **saveptr);
```

Popis

Funkce `strtok` rozděluje řetězec na sekvenci tokenů. Při prvním volání musí parametr *`str`* ukazovat na řetězec který chceme rozdělovat. Funkce si tento řetězec zapamatuje. Při dalších voláních se na místě parametru *`str`* předává `NULL`.

Argument *`delim`*je množina znaků, které oddělují tokeny.

Příklady použití

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define DELIM " ,.-"
int main() {
        FILE *fin;
        int lineno=0;
        char *token, *p, line[80];

        if ((fin = fopen("source", "r")) == NULL) {
                perror("source");
                exit(1);
        }

        while (fgets(line, sizeof(line), fin) != NULL) {
                if ((p = strrchr(line, '\n')) != NULL) *p = '\0';
                printf("%02d: '%s'\n", ++lineno, line);
                token = strtok(line, DELIM);
                while (token != NULL) {
                        //DEBUG:printf("line='%s'\n", line);
                        printf("token='%s'\n", token);
                        token = strtok(NULL, DELIM);
                }
        }
        return 0;
}
```

Odkazy:

- strtok (http://www.cplusplus.com/reference/clibrary/cstring/strtok/)
- `strsep()`
-
-

strspn

Jméno

`strspn, strcspn` — search a string for a set of characters

Přehled

```
#include <string.h>
size_t strspn(const char *s, const char *accept);
size_t strcspn(const char *s, const char *reject);
```

Popis:

Funkce nám řeknou, kolik znaku na počátku řetězce je (`strspn()`) a nebo není (`strcspn()`) z uvedené množiny znaků.

Příklady použití

Funkce mohou být použity například pro přeskočení bílých mezer.

```
#include <stdio.h>
#include <string.h>
int main() {
        int to_skip;

        char *line = " \tvalue, value";
        to_skip = strspn(line, " \t\n");
 printf("%s\n", line+to_skip);

 return 0;
}
```

Nebo pro nalezení konce identifikátoru.

```
#include <stdio.h>
#include <string.h>
#define ID_CHARS "abcdefghijklmnopqrstuvwxyz0123456789"
#define DELIM_CHARS " =,"
int main() {
        char *line = "ab34c=\tvalue, value", *s=line;
 int to_eat;

        to_eat = strcspn(line, "=");
 printf("%s\n", line+to_eat);

 return 0;
}
```

Odkazy:

-
-

strpbrk

Jméno

`strpbrk` — Vyhledání znaků z množiny znaků.

Přehled

```
#include <string.h>
char *strpbrk(const char *s, const char *accept);
```

Popis

Funkce `strpbrk()` vyhledá v řetězci *`s`* první výskyt znaku z množiny znaků *`accept`*.

Vracené hodnoty

Funkce `strpbrk()` frací ukazatel na znak v řetězci *`s`* který odpovídá některému znaku z řetězce *`accept`*. Není li takový znak nalezen, funkce frací `NULL`.

Příklady použití

Odkazy

-
-

strsep

Jméno

`strsep` — extract token from string

Přehled

```
#include <string.h>
char *strsep(char **stringp, const char *delim);
```

Popis

Je li `*stringp` `NULL`, `strsep()` vrací `NULL` a nic nedělá. V opačném případě funkce najde první token v řetězci *stringp. Tokeny jsou odděleny znaky v množnině *`delim`*.

Vracené hodnoty

Příklady použití

Poznámky

Funkce `strsep()` byla vytvořena jako náhrada za funkci `strtok()`, protože tato nemůže zpracovat prázdná pole.

Chyby

Tato funkce má stejné problémy, jako funkce `strtok()`. Jmenovitě modifikuje původní předaný řetězec. Vyhněte se proto jejímu použití.

Odkazy

-
-

strstr

Jméno

`strstr, strcasestr` — hledání řetězce v řetězci

Přehled

```
#include <string.h>
char *strstr(const char *haystack, const char *needle);
#define _GNU_SOURCE
#include <string.h>
char *strcasestr(const char *haystack, const char *needle);
```

Popis

Funkce `strstr()` hledá první výskyt řetězce *`needle`* v řetězci *`haystack`*.

Funkce `strcasestr()` funguje stejně, jen ignoruje velikost písmen.

Vracené hodnoty

Příklady použití

Odkazy

-
-

šablona

Jméno

`šablona` — šablona

Přehled

```
#include <header.h>
char *fce(intarg1, const char *arg2);
```

Popis

Vracené hodnoty

Příklady použití

Odkazy

-
-

Kapitola 84. getopt

- getopt(3) (http://kerneltrap.org/man/linux/man3/getopt.3) by Jeremy on kernel TRAP

Kapitola 85. Meziprocesová komunikace / Interprocess Communiaction

85.1. Pojmenované roury

Odkazy:

- Introduction to Interprocess Communication Using Named Pipes (http://developers.sun.com/solaris/articles/named_pipes.html)

Kapitola 86. Objektově orientované programování v C

Odkazy:

- Object Oriented Programming in C (http://home.comcast.net/~fbui/OOC.html)

Kapitola 87. Extrémní programování v C

87.1. Knihovny pro testování

87.1.1. Knihovna Check

```
$ aptitude show check
Package: check
State: not installed
Version: 0.9.4-3
Priority: optional
Section: devel
Maintainer: Robert Lemmen <robertle@semistable.com>
Uncompressed Size: 332k
Description: unit test framework for C
 Check features a simple interface for defining unit tests, putting little in the way of the developer.
 segmentation faults or other signals. The output from unit tests can be used within source code editors
 IDEs.
```

Do `Makefile` přidáme:

```
TCASES = check_open.o
CFLAGS += -ansi -pedantic -Wall
LDFLAGS += -lcheck

all: test

clean:
 rm -f *.o *~ $(TESTS)

test: run-tests
        @./run-tests

run-tests: run-tests.o $(TCASES)
        $(CC) $(CFLAGS) -o $@ $^ $(LDFLAGS) -lcheck
```

Spouštění testů

Příklad 87-1. `run-tests.c`

```
/* run-tests.c */
#include <stdlib.h>
#include "tests.h"

/*
 * Create Test Suite.
 */
Suite *api_suite(void)
{
        Suite *aSuite = suite_create("API");
```

```
 /* přidáváme postupně jednotlivé testy */
        suite_add_tcase (aSuite, tc_open());

        return aSuite;
}

/*
 * MAIN
 */
int
main(void)
{
        int nf;
        Suite *s = api_suite();
 SRunner *sr = srunner_create(s);

        srunner_run_all(sr, CK_NORMAL);
        nf = srunner_ntests_failed(sr);
        srunner_free(sr);

        return (nf == 0) ? EXIT_SUCCESS : EXIT_FAILURE;
}
```

tests.h

```
#include <check.h>

/* jednotlivé testy */
TCase *tc_open(void)
```

Samotný test je naprogramován v samostatném souboru check_open.c

```
#include "tests.h"
#include <fcntl.h>
...

/*
 *
 */
START_TEST(test_create_file) {
        int fd;

        char msg[] = "File contents.\n";

        fd = open(FILENAME, O_WRONLY|O_CREAT, 0640);
        fail_if(fd < 0, "Can't create file:" FILENAME);
        if (write(fd, msg, strlen(msg)) < 0, "Can't write to file.");
} END_TEST

TCase *tc_open()
{
        TCase *aTCase = tcase_create("open()");

        /* Add all the tests to aTCase */
        tcase_add_test(aTCase, test_create_file);

        return aTCase;
}
```

X. Nástroje

FIXME:napsat.

Kapitola 88. Editory, programy pro editaci textu

Než se začnu věnovat editorům, povíme si jak se bez nich v jednoduchých případech obejít. Řadu jednoduchých úkonů se soubory lze provést jednoduchými příkazy přímo z terminálu, bez nutnosti použít textový editor. Proto dříve něž si popišeme některý z editorů, popíšeme si jak jednoduché úkoly řešit bez jeho pomoci.

Vytvoření prázdného souboru s nulovou délkou

Pro vytvoření prázdného souboru je nejlépe použít příkaz **touch**.

```
$ touch soubor
```

Vytvoření malého jednořádkového souboru

K vytvoření jednořádkového osuboru můžeme použít příkaz shellu **echo**

```
$ echo "jedna řádka textu" >soubor
```

Vytvoření několikařádkového souboru jedním příkazem **echo**

Příkaz echo může použít také pro vytvoření několikařádkového souboru. Přepínač `-e` nám umožní použít znak konce řádku `'\n'`. Můžeme tedy v jednom příkaze napsat několik řádků.

```
$ echo -e "první řádek\ndruhý řádek\ntřetí řádek" > soubor
```

Vytvoření malého souboru příkazem **cat**

Dalším jednoduchým způsobem jak vytvořit malý několikařádkový soubor je pomocí programu **cat**

```
$ cat ->soubor
Toto je malý pokusný soubor
druhý řádek pokusného souboru.
Ctrl-d
```

nebo

```
$ cat ->soubor <<EOFToto je malý pokusný soubor
druhý řádek pokusného souboru.
EOF
```

Rozšíření souboru přidáním řádku na konec příkazem **echo**

Můžeme použí příkaz echo a přidat na konec souboru jednu řádku. Takto můžeme přidávat řádky dle potřeby.

```
$ echo "řádek textu" >>soubor
```

Rozšíření souboru přidáním textu na konec příkazem **cat**

```
$ cat ->>soubor
další řádek textu
a ještě jeden
Ctrl-d
```

nebo

```
$ cat ->>soubor <<EOF
další řádek textu
a ještě jeden
EOF
```

Velikost pořizovaného souboru je omezena jen naší schopností psát bezchybně. V případě použití příkazu **echo** sice můžeme editovat příkaz před tím něž jej spustíme, ale jednou přidané řádky nelze odebrat. U příkazu **cat** je to obdobné s tím že jediná možnost editace na řádku je tlačítko **Backspace**, tedy smazání posledně napsaného znaku a to jen na témže řádku. K předchozím řádkům se taktéž vrátit nemůžeme.

Uvedené příklady jsou sice značně primitivní ale dají se s úspěchem používat. S pomocí dalších příkazů a jejich kombinací lze provádět další složitější úkony, ale tady je použití editoru jednodušší.

88.1. ed

ed je jedním z prapůvodních editorů jenž se objevil v prvopočátcích UNIXu. Budu-li mít někdy čas a dobrou vůli, podívám se do historických záznamů a popíši editory a jejich vývoj v dávných dobách. Zde uvádím **ed** zejména z nostalgických důvodů, ale taky proto, že v případě omezených zdrojů, zejména rychlosti terminálu nám může být velmi užitečný. Všechny další editory které zmíním jsou totiž editory celostránkové. Z toho mimo jiné vyplývá že při spuštění editoru a natažení soubor zobrazí tento na celé stránce terminálu, což v případě velmi omezené rychlosti terminálu trvá až nepříjemně dlouho. Ano je to vyjíměčný případ, ale právě pro tento případ je **ed** vhodný. Přenáší totiž, jak dále uvidíme, jen opravdu nezbytné minimum znaků. Důvod proč je **ed** takto nenáročný tkví právě v historii. V té době:

- nebyly obrazovkové terminály tak rozšířené
- nikdy jste nemohli vědět jestli v místě kde budete pracovat nějaký volný mají
- modemové linky byly pomalé, často byly schopné přenést jen několik málo znaků až desítek znaků za sekundu, podle rychlosti
- na řadě míst byl přístupný jen terminál typu *elektrický psací stroj*

Značně nízké nároky editoru **ed** na přenosovou rychlost a vlastnosti terminálu nám proto mohu pomoci vyřešit mezní situace s omezenými zdroji.

Instalace je jednoduchá, nic neinstalujeme. **ed** patří k základní vybavě všech UNIX-like systémů se kterými jsem se potkal.

```
$ ed novy-soubor
```

Struktura příkazu vypadá následovně:

[adresa [,adresa]]příkaz[parametry]

První dva nepovinné parametry jsou počáteční a koncová adresa které se oprace týká. Adresovány jsou celé řádky. Poté následuje příkaz za kterým mohou být uvedeny parametry příkazu, ty se liší příkaz od příkazu.

Tabulka 88-1. Přehled adresních příkazů

adresa	popis
.	aktuální řádek
$	poslední řádek v bufferu
číslo n	n-tý řádek v bufferu, n musí být v rozdahu [0,$]
- nebo ^	předchozí řádek, je stejné jako -1 a může se opakovat čímž se kumuluje
-n nebo **^n**	n-tý předchozí řádek
+	další řádek, je stejné jako +1
+n nebo **_n**	n-tý předchozí řádek

adresa	popis
, nebo **%**	celý buffer, je to stejné jako **1,$**
;	od aktuálního řádku do konce, stejné jako **.,$**
/re/	Další řádek obsahující regulární výraz re. Pokud vyhledávání dojde konce bufferu, začíná od začátku. // opakuje poslední hledání.
?re?	Předchozí řádek obsahující regulární výraz re. Je stejné jako /re/ ale hledá opačným směrem. ?? opakuje poslední hledání.
'x	Řádek dříve označené označený malým pismenem. x je malé písmeno.

Tabulka 88-2. Přehled příkazů

příkaz	název	popis
a	append	následující text/řádky budou vloženy za řádek specifikovaný v příkaze **a**
c	chnage	nahradí/přepíše řádky uvedené v příkazu **c** následujícím textem
d	delete	řádky specifiované v příkazu **d** jsou smazány/odstraněny
e file	edit	
e !command	edit	
E	edit	
f file		
g/re/command-list		
G/re/		
H		zapíná/vypíná popis chyb
h		vytiskne popis poslední chyby
i	insert	následující text/řádky budou vloženy před řádek specifikovaný v příkaze **i**
j		spojí řádky do jednoho
k*lc*		označí řádek malým písmenem *lc*
l	list	
m	move	
q	quit	ukončení práce editoru
s	substitute	příkaz provede vyhledání řetězce a nahrazení jiným řetězcem
w	write	zapíše editovaný soubor na disk
P		zapíná a vypíná command prompt

* *Další příkazy: G, V, f, l, t, !!*

FIXME: následují nezpracované texty.

```
radek@yoda:~$ which ed
/bin/ed
radek@yoda:~$ dpkg -S ed
```

,s/old/new/g

88.2. vi

Editory rodiny vi

Vi je jeden ze starších editorů, nepřímý následník editoru 88.1. Jako takový v sobě zahrnuje i příkazy tohoto editoru. V češtině bylo popsáno několik návodů na které bych čtenáře rád odkázal. Jedná se zejména o velmi čtivý dokument vim skutečný editor textů (http://www.kit.vslib.cz/~satrapa/docs/vim/index.htm) od Pavla Satrapy (http://www.kit.vslib.cz/~satrapa/omne/). Ano, dokument popisuje vim který je rozšířením editoru vi.

Další stránky/odkazy:

- Vim (http://www.penguin.cz/~vim)
- Vim.ORG (http://www.wim.org/)

Příkazy, ovládání, …

i - insert
I - insert na začátek řádku
a - append
A - append na konec řádku
x - delete character, pod kurzorem
r - replace, přepsání zaku na pozici kurzoru
R - přepnutí do přepisovacího režimu od pozice kurzoru v četně
J - spojení dvou řádků

u - undo

y - yank, kopíruje řádky do bufferu
p - vloží obsah bufferu za následující řádek
v,V,Ctrl-v - visual - označení bloku

Navigace po souboru

h - ←
j - dolů
k - nahorů
l - →
H - top of the screen
G - end of file - přesun na konec souboru
g nebo gg - přesun na začátek souboru
L - bottom of the screen

Tabulka 88-3. Navigace po souboru

příkaz	popis
*n*G	skok na řádek číslo n

/,? - vyhledávání (n - další (next), N - předchozí (previous)

ZZ - quit

:1,$
:.,$
:%

:% s/\<file\>/soubor/g

:syntax-on
:set number
:set all vypíše všechna nastavení

:r !date

```
# vim -d /etc/passwd /etc/passwd-
```

Konfigurační soubory editoru vim jsou:

- `/etc/vim/vimrc` — konfigurace společná pro všechny uživatele
- `~/.vimrc` — konfigurace uživatele

Příkazy:

- **:syntax on** — barevná syntaxe

88.3. emacs

Zdroje, odkazy
- http://www.emacswiki.org
- Sams Teach Yourself Emacs in 24 Hours (http://www.emacs.uniyar.ac.ru/doc/em24h/index.htm)

88.3.1. Instalace

V Debian Sarge.

Na vybranou máme několik balíčků. `emacs`, `emacs21`, `emacs21-nox` a `xemacs21`.

```
# aptitude install emacs21-nox
```

88.3.2. Národní znaky / čeština v Emacsu

Do souboru `$HOME/.Xresources` přidáme řádku

```
emacs*Font: -misc-fixed-medium-r-normal--18-*-iso8859-2
```

pro jistotu vytvoříme soubor `$HOME/.Xdefaults` jako symbolický odkaz na `.Xresources`

```
$ cd
$ ln -s .Xresources .Xdefaults
```

Spouštění emacsu na vzdáleném stroji sebou obnáší jednu záludnost. Tou je, že emacs nemá přístup ke klávesnici. Aby nám v pořádku chodila čeština, potřebujeme mít jistotu že celá cesta je *8-bit clean* a na svém počítači v pořádku zobrazování českých znaků i českou klávesnici. Poslední co potřebujeme vysvetlit emacsu je že na vstupu nemá používat žádné transformace ani překlady znaků, protože k němu přichází již čistá čeština v iso-8859-2. Toho dosáhneme nastavením prázdného vstupního módu. Například tak že do souboru `.emacs` na vzdáleném stroji vložíme řádky:

```
; Nastavení pro české znaky přes ssh
(standard-display-european 2)
(set-input-mode nil nil 1)
```

88.3.3. Version Control

Tabulka 88-4. Emacs Version Control

C-x v i	vc-register	registrace nového souboru do systému správy verzí
C-x C-q	vc-toggle-read-only	check in / check out - podle aktuálního stavu
C-x v v	vc-next-action	Stejné jako u C-x C-q. Check in/out - podle aktuálního stavu
C-x v ~	vc-version-other-window	Pohled na další (jiné) revize souboru.
C-x v =	vc-diff	Změny (rozdíly) mezi revizemi.
C-x v u	vc-revert-buffer	undo checkout
C-x v c	vc-cancel-version	odstranění poslední revize
C-x v d	vc-directory	seznam souborů které nejsou aktuální (*up to date*)
C-x v g	vc-annotate	ukáže kdy byla která řádka souboru přidána a kým
C-x v s	vc-create-snapshoot	označení všech souborů (aktuálních revizí) symbolickým jménem
C-x v r	vc-retrieve-snapshoot	undo checkoutes a návrat ke snapshootu s daným jménem
C-x v l	vc-print-log	prohlížení deníku
C-x v a	`vc-update-change-log`	aktualizace souboru `ChangeLog`. Otevře soubor `ChangeLog` v aktuálním adresáři, přesněji v kořenovém adresáři projektu a vytvoří nové záznamy pro nové verze souborů od poslední aktualizace souboru `ChangeLog`. Tento příkaz pracuje jen s RCS nebo CVS, nepracuje s SCCS.
C-u C-x v a		Funguje jako příkaz `C-x v a`, ale jen pro soubor v aktuálním bufferu.
M-1 C-x v a		Funguje podobně jako předchozí příkaz `C-x v a` pro všechny navštívené soubory které jsou udržovány v RCS. Vloží všechny záznamy do deníku pro akruální adresář.
C-x v m	vc-merge	
C-x v h	vc-insert-headers	

88.3.4. Deník změn (*ChangeLog*)

* *section id="emacs.changelog"*

Zdroje a odkazy:

- Rcs, a GNU Emacs RCS interface (http://www.uvm.edu/~ashawley/rcs/manual/gnu_emacs/rcs_1.html)
- http://www.cs.utah.edu/dept/old/texinfo/emacs18/emacs_25.htmlEditing Programs

Deník změn je soubor pojmenovaný `ChangeLog` jenž obsahuje záznamy o změnách v souborech projektu. Tyto záznamy chronologicky řazené popisují kdo a kdy provedl změnu a v jakých souborech.

Na konec souboru `ChangeLog` můžete přidat *copyright notice* a *permission notice*. Například

> Copyright 1997, 1998 Free Software Foundation, Inc.
> Copying and distribution of this file, with or without modification,
> are permitted provided the copyright notice and this notice are
> preserved.

Záznam v deníku začíná řádkem hlavičky který obsahuje aktuální datum, jméno autora změny a jeho emailovou adresu. Každý řádek (další) mimo počáteční řádky záznamů začíná mezerou nebo tabelátorem.

> 1993-05-25 Richard Stallman <rms@gnu.org>
>
> * man.el: Rename symbols ‘man-*’ to ‘Man-*’.
> (manual-entry): Make prompt string clearer.
>
> * simple.el (blink-matching-paren-distance):
> Change default to 12,000.
>
> 1993-05-24 Richard Stallman <rms@gnu.org>
>
> * vc.el (minor-mode-map-alist): Don’t use it if it’s void.
> (vc-cancel-version): Doc fix.

Jeden záznam popisuje několik změn, každá ná dalším řádku.

C-j

FIXME:

M-x change-log-merge

FIXME:

Ctrl+**x [** `backward-page`

FIXME:

Ctrl+**x]** `forward--page`

FIXME:

Meta+**[** `backward-paragraph`

FIXME:

Meta+**]** `forward-paragraph`

FIXME:

Esc h `mark-paragraph`

FIXME:

Ctrl+**x a**
`add-change-log-entry`

Pomáhá udržovat záznamy o provedených změnách v programu/souboru. Předpokládá(funkce) že máme k dispozici soubor do kterého provádíme chronologicky záznamy popisující individuální změny.

Implicitní jméno tohoto souboru je `ChangeLog` a je uložen/očekáván ve stejném adresáři jako editované soubory.

Nalezne soubor změn `ChangeLog` a přidá do něj záznam pro dnešní den, soubor a **FIXME:**defun.

Ctrl+x 4 a
`add-change-log-entry-other-window`

Příkaz vloží novou položku/záznam pro aktuálně editovaný soubor do deníku změn, souboru `ChangeLog`.

Je-li parametr/proměnná `add-log-keep-changes-together` ne `nil`, příkaz vloží záznam do aktuálního raději než aby otevíral nový záznam.

Je-li parametr/proměnná `change-log-version-info-enabled` ne `nil`, příkaz přídá do hlavičky číslo verze. Tot číslo získá prozkoumáním první desetiny souboru (10%) s použitím regulárního výrazu v proměnné `change-log-version-number-regexp-list`.

`add-change-log-entry-rcs-format`

Default: `",v %s"`

88.3.5. Editace souborů Debianu

* *section id="emacs.debian" condition="author"*

Zdroje a odkazy:

- **FIXME:**

Nainstalujeme si balíček maker pro Emacs `dpkg-dev-el`

```
# apt-get install dpkg-dev-el
```

```
Where is debian-changelog-mode.el?

/etc/emacs/site-start.d$ cat 50dpkg-dev.el
[..]
(autoload 'debian-changelog-mode "/usr/lib/emacs/site-lisp/debian-changelog-mode.el"

while I at the same time see:

$ zgrep debian-changelog Contents-i386.gz
$ ls -al Contents-i386.gz ; md5sum Contents-i386.gz
-rw-r--r--   1 joost    users      993611 Feb 11 17:42 Contents-i386.gz
ee61bca26b3bfc95c4ffa01b42567d9d  Contents-i386.gz
```

```
Also, in my ~/.emacs I've got things like

  (define-key global-map [home] 'beginning-of-line)

and

  (global-font-lock-mode t)
```

```
These also stopped working since my last update.
Anybody know why?

Thanks!


$ dpkg -l dpkg dpkg-dev 'emacs*'
ii  dpkg            1.4.0.20       Package maintenance system for Debian Linux
ii  dpkg-dev        1.4.0.20       Package building tools for Debian Linux
ii  emacs           19.34-13       The GNU Emacs editor.
ii  emacs-el        19.34-13       GNU Emacs LISP (.el) files.
ii  emacs-lisp-intr 1.04-1         Programming in Emacs Lisp: An Introduction



bbdb (2.34.cvs20020418-2) unstable; urgency=high

  * Urgency=high because the Bugfix for #143463 should really go into
    woody. I have not made any other changes in the emacsen-install files,
    and there are no Bug Reports since my last Upload, so i think this is OK.
  * I now generate some html Pages from the texinfo Source and install the
    doc-base things. So you can read the bbdb documentation from your
    Webbrowser if you want that.

 -- Joerg Jaspert <joerg@debian.org>  Sun, 21 Apr 2002 20:02:31 +0200

bbdb (2.34.cvs20020418-1) unstable; urgency=low

  * New Upstream release. This fixes bbdb-gui which (closes: #143463)
  * Changed my Email Address.

 -- Joerg Jaspert <joerg@debian.org>  Thu, 18 Apr 2002 20:53:56 +0200
```

Klávesové zkratky a příkazy

Ctrl+c Ctrl+v

FIXME: New Version

Ctrl+c Ctrl+a

FIXME: Add Entry

Ctrl+c Ctrl+b

FIXME: Close Bug

Ctrl+c Ctrl+e

FIXME: Unfinalise

Ctrl+c Ctrl+f

FIXME: Finalise

Ctrl+c Ctrl+c

FIXME: Finalise+Save

FIXME:

88.3.6. Emacs jako server

Emacs se dá provozovat v režimu server. Takto spuštěný Emacs reaguje ma žádost klienta emacsclient o editaci souboru. To je výhodné v tom, že „editace“ souboru přes emacsclient nemusí spouštět instanci emacsu, ale jen v tom běžícím otevře nový buffer. Tedy ušetříme paměť, a taky čas nutný ke spuštění nové instance emacsu.

Emacs server spouštíme přes `(server-start)` v souboru `~/.emacs`, nebo přímo voláním funkce přes `M-x server-start`

Tabulka 88-5. Emacs server

M-x server-start	server-start	spuštění emacs serveru
C-x #		uzavření bufferu otevřeného přes emacsclient

88.3.7. PSGML

Editace SGML a XML dokumentů.

Pro bližší informace si můžeme spustit nápovědu

```
$ info psgml
```

Tabulka 88-6. příkazy

	sgml-describe-entity	Describe the properties of an entity as declared in the current DTD.
Ctrl+c Enter	**sgml-split-element**	Split the current element at point. If repeated, the containing element will be split before the beginning of then current element. Typical use is to start a new paragraph element when inside a paragraph.
Esc Ctrl+k	**sgml-kill-element**	

88.3.8. Klávesové skratky a příkazy

Tabulka 88-7. Pohyb po textu

`C-b`	backward-char	Move left one character (backwards)
`C-f`	forward-char	Move right one character (forwards)
`C-p`	previous-line	Move up one line (previous)

`C-n`	next-line	Move down one line (next)
`C-v`		Move forwards one screenful
`M-v`		Move backwards one screenful
`C-l`		Refresh screen and recenter on current line
`M-f`		Move forwards one word
`M-b`		Move backwards one word
`C-a`	beginning-of-line	
`C-e`	end-of-line	
`M-a`		Move to the beginning of the sentence
`M-b`		Move to the end of the sentence

Tabulka 88-8. Stříhání a kopírování

C-space		označení jednoho konce textu
C-x C-x	exchange-point-and-mark	prohození počátečního a koncového bodu označeného textu
C-w	kill-region	
M-w		

Tabulka 88-9. Práce se soubory

zkratka	příkaz	popis
C-x C-b		přepnutí do jiného bufferu - budeme vyzváni k zadání názvu bufferu
C-x C-f	**find-file**	otevření souboru, taktéž menu *Files / Open File*
C-x C-k	kill-buffer	odstranění aktuálního bufferu
C-x C-s		uložení souboru
C-x C-c		ukončení práce s editorem
C-x C-w		uložení souboru pod jiným jménem

Tabulka 88-10. Práce s buffery a okny

C-x b *`jméno_bufferu`*		vytvoření nového bufferu, nebo přepnutí do existujícího
C-x C-b		zobrazení seznamu otevřených bufferů
C-x k *`jméno_bufferu`*		zrušení existujícího bufferu
M-x rename-buffer	rename-buffer	přejmenování bufferu
C-x 2		rozdělení okna na další dvě, vodorovné dělení
C-x 3		rozdělení okna na další dvě, svislé dělení
C-x o		přepínání mezi okny

C-x 0		zrušení ktivního okna
C-x 1		aktivní okno přes celou plochu, zrušení ostatních oken

Tabulka 88-11. SGML

C-M-a		
C-M-e		
C-M-d	sgml-down-element	
C-M-b	sgml-backward-element	
C-M-f	sgml-forward-element	
C-c <	sgml-insert-start-tag	
C-c /	sgml-insert-end-tag	
C-c C-o	sgml-next-trouble-spot	
Ctrl+c Ctrl+e	**sgml-insert-element**	insert element
Ctrl+c Ctrl+r	**sgml-tag-region**	
C-c C-q	sgml-fill-element	

Tabulka 88-12. Různé

M-x příkaz		přímé vyvolání příkazu podle jména
C-x C-o	delete-blank-lines	
C-x m	compose-mail	
C-z		Suspend Emacs
C-x C-c		Quit Emacs
Esc ! *příkaz*		Vykonání shell příkazu
Esc ! *příkaz*		Vykonání shell příkazu

88.3.9. Módy

FIXME: Tato část popisuje módy do nichž lze Emacs po „nainstalování“ příslušných „programových“ souborů přepnout. Či jež leze jinak využívat.

Krátký přehled některých módů

IDO

FIXME:`ido.el` (http://www.cua.dk/ido.html) — IDO mode

```
C-x b -- switch to buffer
C-x f -- open file
```

mód

FIXME:

XI. Práce s textem a zpracování dokumentace

FIXME:napsat.

Kapitola 89. DocBook

Poznámky k psaní dokumentace v DocBook

89.1. Emacs nXML

Odkazy:

- nXML mode (http://www.thaiopensource.com/nxml-mode/) homepage
- James Clark unveils a new XML mode for GNU Emacs (http://www.xmlhack.com/read.php?item=2061)
- NxmlMode (http://www.emacswiki.org/cgi-bin/wiki/NxmlMode) on Emacs Wiki

Tabulka 89-1. Přehled klávesových zkratek

zkratka	funkce	význam
C-return	`nxml-complete`	auto complete
/	`nxml-electric-slash`	
S-mouse-2	`nxml-mouse-hide-direct-text-content`	
C-c C-n	`rng-next-error`	next trouble spot
C-c C-v	`rng-validate-mode`	
C-c C-u	`nxml-insert-named-char`	
C-c C-d	`nxml-dynamic-markup-word`	
C-c C-x	`nxml-insert-xml-declaration`	
C-c TAB	`nxml-balanced-close-start-tag-inline`	
C-c C-b	`nxml-balanced-close-start-tag-block`	
C-c RET	`nxml-split-element`	
C-c C-f	`nxml-finish-element`	
ESC h	`nxml-mark-paragraph`	
ESC }	`nxml-forward-paragraph`	
ESC {	`nxml-backward-paragraph`	
ESC C-p	`nxml-backward-element`	
ESC C-n	`nxml-forward-element`	
ESC C-d	`nxml-down-element`	
ESC C-u	`nxml-backward-up-element`	
C-c C-s C-t	`rng-set-document-typa-and-validate`	
C-c C-s C-l	`rng-save-schema-location`	

zkratka	funkce	význam
C-c C-s C-f	`rng-set-schema-file-and-validate`	
C-c C-s C-a	`rng-auto-set-schema-and-validate`	
C-c C-s C-w	`rng-what-schema`	
C-c C-o C-o	`nxml-hide-other`	
C-c C-o TAB	`nxml-show-direct-subheadings`	
C-c C-o C-l	`nxml-hide-text-content`	
C-c C-o C-k	`nxml-show-subheadings`	
C-c C-o C-s	`nxml-show`	
C-c C-o C-d	`nxml-hide-subheadings`	
C-c C-o C-e	`nxml-show-direct-text-content`	
C-c C-o C-c	`nxml-hide-direct-text-content`	
C-c C-o C-r	`nxml-refresh-outline`	
C-c C-o C-t	`nxml-hide-all-text-content`	
C-c C-o C-a	`nxml-show-all`	

Tabulka 89-2. Doporučené funkce

zkratka	funkce	význam
C-M-\	`indent-region`	
M-q	`fill-paragraph`	
C-x r k	`kill-rectangle`	

89.2. DocBook Wiki

89.2.1. Instalace v Debian Lenny

Není nic jednoduššího:

```
# aptitude install docbookwiki
```

FIXME: Ověřit: Teda je třeba mít ve zdrojích contrib a non-free.

Po instalaci je třeba restartovat www server.

```
# /etc/init.d/apache2 restart
```

A nyní se můžeme podívat na výsledek instalace `http://server/books/`

89.2.2. Různé postupy

89.2.2.1. Vygenerování výstupních formátů

Tato úloha se děje jednou denně, jak je vidět v souboru `/etc/cron.d/docbookwiki`. Pokud potřebujeme přegenerovat výstupy ihned, pomužem si příkazem:

```
# sudo -u dbwiki /usr/share/docbookwiki/content/downloads/make-all-downloads.sh
```

89.2.2.2. Import existujícího DocBook XML dokumentu

```
/usr/share/docbookwiki/content# ./import.sh /root/dbwiki/docbook.xml docbook cs_CZ.UTF-8
```

Kapitola 90. CMS

A content management system (CMS) is computer software used to create, edit, manage, and publish content in a consistently organized fashion.[1] CMSs are frequently used for storing, controlling, versioning, and publishing industry-specific documentation such as news articles, operators' manuals, technical manuals, sales guides, and marketing brochures. The content managed may include computer files, image media, audio files, electronic documents, and Web content.

Wikipedia (http://en.wikipedia.org/wiki/Content_management_system)

* *MARK*

A content management system (CMS) is computer software used to create, edit, manage, and publish content in a consistently organized fashion.[1] CMSs are frequently used for storing, controlling, versioning, and publishing industry-specific documentation such as news articles, operators' manuals, technical manuals, sales guides, and marketing brochures. The content managed may include computer files, image media, audio files, electronic documents, and Web content.

—Wikipedia (http://en.wikipedia.org/wiki/Content_management_system)

90.1. Geeklog

Odkazy:

- www.Geeklog.net (http://www.geeklog.net)
-
-

90.1.1. Instalace

Proberu instalaci na čistém systému, pokud jsem na něco nezapoměl. Geeklog instaluji jako jedinou aplikaci, ale umístňuji ji do vlastního adresáře abych ponechal prostor pro další aplikace. Geeklog tedy **není** umístněn do kořenu webu.

```
# aptitude install apache2
# aptitude install libapache2-mod-php5
# aptitude install mysql-server-5.0
# aptitude install php5-mysql
```

Instalace dále vyžaduje funkční smtp server. Proužil jsem pro jednoduchost ssmtpd, protože na instalovaném počítači není třeba složitejšího poštovního serveru.

Máme tedy nainstalovány balíčky s potřebnými programy a knihovnami, a stáhneme a připravíme si samotný Geeklog. Z webu Geeklogu (http://www.geeklog.net) jsem stáhle balíček `geeklog-1.5.0.tar.gz` a umístnil jej do adresáře `/usr/local/download`. První věc kterou jsem udělal, bylo rozbalení balíčku.

```
# cd /usr/local/share
# tar xzvf /usr/local/download/geeklog-1.5.0.tar.gz
```

Rozbaleným souborům a adresářům nastavíme správného vlastníka, což je v našem případe `www-data`, provozovatel vebového serveru.

```
# chown -R www-data:www-data /usr/local/share/geeklog-1.5.0
```

Můžeme rovněž rovnou všechny soubory zabezpečit před ostatními uživateli systému. Toto vřele doporučuji.

```
# chmod go-rwx /usr/local/share/geeklog-1.5.0
```

Před další konfigurací geeklogu, která je velmi jednoduchá, si musíme připravit apache a mysql server. Začnu apache. Jediné co je třeba učinit je vysvětlit apache kde se nachází geeklog a kde bude umístněn na webu. To provedem vsunutím následujících řádků do souboru default webu `/etc/apache2/sites-available/default`. Do sekce `<Virtual Host *>`, která je v tomto souboru jen jedna přidáme řádky:

```
# Geeklog
Alias /geeklog /usr/local/share/geeklog-1.5.0/public_html
```

Po reloadu apache ten tedy bude vědět kde se adresář s geeklogem nachází a ten bude vystaven na webu jako *`tento-server`*`/geeklog`. Ještě připravíme MySQL databázi. Nastavíme administrátora MySQL a přihlásíme se do databáze. Následující ukázka provádí postupně vytvoření databáze, vytvoření uživatele a přidělení dostatečných práv tomuto uživateli.

```
# mysql -u root -p
Enter password:
...
mysql> CREATE DATABASE geeklog DEFAULT CHARACTER SET = 'utf8';
mysql> CREATE USER 'geek'@'localhost' IDENTIFIED BY 'tajne-heslo';
mysql> GRANT ALL PRIVILEGES ON geeklog.* TO 'geek'@'localhost';
```

Vytvořená databáze se jmenuje `geeklog`, uživatel jenž k ní má práva se pak jmenuje `geek` a jeho heslo je `tajne-heslo`. Že je vše v pořádku si odzkoušíme tak že se pod tímto uživatelem přihlásíme do databáze.

```
# mysql --user=geek --pass=tajne-heslo geeklog
```

Jestli vše funguje, můžeme přistoupit k instalaci geeklogu. To učiníme tak, že svůj webový prohlížeč obrátíme k adrese `http://`*`server-na-který-jsme-instalovali`*`/geeklog/admin/install/index.php`, a řídíme se pokyny na zobrazené stránce.

XII. Různé

Tato část obashuje různé zatím nezařazené věci. Také jsou zde texty importované z původní verze ktéto knihy o kterou jsem při havárii disku přišel.

Kapitola 91. HTML

91.1. Seznamy

* ***To be done:***

91.1.1. Seznam s odrážkami

* ***To be done:***

```
<ul>
  <li>  </li>
  ...
</ul>
```

Kapitola 92. CSS

* ***To be done:***

92.1. Menu ze seznamu

Pro menu ze seznamu použijeme 91.1.1. Tento však uzavřeme do `div` elementu.

Příklad 92-1. HTML kód menu ze seznamu

```
<div class="hmenu">
  <ul>
    <li><a href="...">První</a></li>
    <li><a href="...">Aktuální</a></li>
    <li><a href="...">Poslední</a></li>
  </ul>
</div>
```

Příklad 92-2. CSS kód pro menu ze seznamu

```
/* Horizontal menu. */
div.hmenu {
        width: 100%;
}
div.hmenu ul {
        list-style: none;
}
div.hmenu ul li {
        float: left;
        margin: 0px 3px;
        padding: 2px 4px;
        background: #E0E0FF;
}
```

* ***To be done:***

Kapitola 93. Základy UNIXu

Základy užití

93.1. Přihlášení a odhlášení

Přistoupíme-li k nepřihlášené konsole unixu, uvidíme na obrazovce podpbnou výzvu

```
Debian GNU/Linux 3.0 joshua tty2

joshua login:
```

jedná se o výzvu k přihlášení. Nejdříve zadáme své přihlašovací jméno (login) a pak nás přihlašovací program, který se shodou okolností jmenuje také login vyzve k prokázání naší totožnosti (autentikaci).

* *FIXME: autentikace/autorizace - vyjasnit si pojmy, opravit*

Svou totožnost prokazujeme znalostí tajného hesla.

```
Debian GNU/Linux 3.0 joshua tty2

joshua login: radek
Password: zadávané heslo se nezobrazuje
Last login: thu Feb 14 17:33:38 2002 from kvark.firma.cz on ssh
Linux joshua 2.2.20 #1 Wed Jan 23 15:14:58 CET 2002 i686 unknown

....
You have new mail.
radek@joshua:~$
```

93.2. Soubory a adresáře

Popis stromové adresářové struktury.

93.3. Maska, divoké znaky

FIXME:

93.4. Vstup a výstup (I/O)

>file	přesměrování výstupu do souboru, soubor je vytvořen
>>file	přesměrování výstupu, výstup je připojen na konec souboru

<file	přesměrování vstupu
<<EOT .... EOT	přesměrováni vstupu
2>&1	přesměrování proudu `stderr` do proudu `stdin`

Kapitola 94. Připojení z počítačů Apple

Odkazy a zdroje:
- Odkaz (http://www.cosi.org)

94.1. Appleshare

* *Zrušit tuto sekci.*

Odkazy a zdroje:
- netatalk (http://www.umich.edu/~rsug/netatalk)

Jako AppleShare server můžeme použít program netatalk.

94.1.1. Konfigurace

Nejdříve vytvoříme server záznamem v souboru `/etc/netatalk/afpd.conf`

```
- -ipaddr 10.16.66.20
```

Jednotlivé adresáře exportujeme ven záznamy v souboru `/etc/netatalk/AppleVolumes.default`. Na každém řádku je uvedeno

`cesta_k_adresáři` "*`exportovaný_název`*"

Například

```
~/                      "Home Directory"
/bd/2/library/          "Knihovna"
```

94.2. Appletalk

* *Zrušit tuto sekci.*

Pro podporu protokolu Appletalk je třeba nakonfigurovat jádro. Je potřeba povolit `AF_APPLETALK`. Při konfiguraci v menu Networking options zaškrtnout volbu `Appletalk DDP`.

94.3. netatalk

Program **netatalk** slouží jak server jenž poskytuje různé služby počítačům Apple Macintosh v síti.

K 2004-04-05 je v Debian Woody balíček *netatalk* dostupný v těchto verzích:

- stable: `1.5.3.1-1`
- testing: `1.6.4-1`

Verze 1.5.3 jenž mám nasazenu má problémy s kódováním češtiny.

94.3.1. Verze 1.5.3.1-1 z Woody

Instalace z binátního balíčku

```
# apt-get install netatalk
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  netatalk
0 packages upgraded, 1 newly installed, 0 to remove and 4  not upgraded.
Need to get 348kB of archives. After unpacking 773kB will be used.
Get:1 http://debian woody/non-US/main netatalk 1.5.3.1-1 [348kB]
Fetched 348kB in 1s (318kB/s)
Instaluji balík netatalk.
(Čtu databázi ... nyní je nainstalováno 33702 souborů a adresářů.)
Rozbaluji netatalk (z .../netatalk_1.5.3.1-1_i386.deb) ...
Nastavuji balík netatalk (1.5.3.1-1) ...
Starting AppleTalk services (this will take a while):  atalkd afpd papd.
```

Konfigurace sestává z úpravy souborů v adresáři `/etc/netatalk` Nejdříve nakonfigurujeme afpd. Jeho konfigurace je v souboru `/etc/netatalk/afpd.conf`

```
mesic -ipaddr 10.16.66.20 -uamlist uams_pam.so
```

poté napíšeme exporty adresářů do souboru `/etc/apt/AppleVolumes.default`

```
~/              "Home Directory"
/bd/3/library/  "Knihovna"
```

a můžeme server spustit

```
# /etc/init.d/netatalk restart
Restarting AppleTalk Daemons (this will take a while)..done.
```

start trvá trochu déle. V deníku se objevi zprávy

```
Apr  7 16:09:29 moon afpd[2551]: mesic:AFPServer@* started on 65280.33:128 (1.5.3.1)
Apr  7 16:09:29 moon afpd[2551]: ASIP started on 10.16.66.20:548(2) (1.5.3.1)
Apr  7 16:09:29 moon afpd[2551]: uam: loading (/usr/lib/netatalk/uams_pam.so)
Apr  7 16:09:29 moon afpd[2551]: uam: uams_pam.so loaded
Apr  7 16:09:29 moon afpd[2551]: uam: "Cleartxt Passwrd" available
```

94.3.2. Instalace

Netatalk můžeme instalovat standardně pomocí apt.

```
# apt-get install netatalk
```

Jak jsem již zmínil, tato verze mi nevyhovuje a tak bych rád nainstaloval novější. V backports.org ("http://www.backports.org) zatím není a tak mi nezbývá než si jej zkompilovat sám. Zdrojový balíček ze Sarge má závislosti na Sarge a tak jsem přistoupil k překladu ze zdrojů.

```
# mkdir -p /usr/src/deb/netatalk
# cd /usr/src/deb/netatalk
# apt-get -t sarge source netatalk
# apt-get install libdb3-dev
```

```
# cd netatalk-1.6.4
# ./configure --prefix=/opt/netatalk-1.6.4 --with-shadow --enable-fhs \
            --with-tcpwrappers --with-mangling --enable-timelord \
            --enable-overwrite --with-pkgconfdir=/etc/netatalk-1.6.4 \
            --with-nls-dir=/opt/netatalk-1.6.4/nls --without-ssl-dir \
            --disable-logger --without-logger
# make
```

94.3.3. Konfigurace

/etc/netatalk/afpd.conf

```
mesic -ipaddr 10.16.66.20
```

/etc/netatalk/AppleVolumes.default

```
/bd/3/library "Knihovna"
```

/etc/netatalk/netatalk.conf

```
# egrep -v "^#|^[ \t]*$" netatalk.conf
AFPD_MAX_CLIENTS=20
ATALK_NAME=‘echo ${HOSTNAME}|cut -d. -f1‘
AFPD_GUEST=nobody
ATALKD_RUN=yes
PAPD_RUN=yes
AFPD_RUN=yes
TIMELORD_RUN=no
ATALK_BGROUND=no
```

Kapitola 95. Bluetooth

Odkazy a zdroje:

- Oficiální stránky standardu Bluetooth (http://www.bluetooth.org)
- BlueZ (http://sourceforge.net/projects/bluez)
- Sbírka odkazů na téma Bluetooth a Linux (http://www.holtmann.org/linux/bluetooth)
- Linux a Bluetooth? Žádný problém! (návod) (http://mobil.idnes.cz/mobilni_komunikace/mobilni_technologie/bluetooth/blue
- Linux a Bluetooth klávesnice a myšky (http://www.root.cz/clanek/1948)
- NOKIA 6310i, Bluetooth, GPRS od T-mobile CZ a Linux (http://www.utia.cas.cz/user_data/zajicek/misc/D_Link_6310i_T-mobile.html)

ToDo

1. Posbírat sdostatek informací.
2. Rozchodit BT na scém notebooku `yoda`.
3. Rozchodit připojení BT zařízení. Telefon NOKIA 6310.

95.1. Sprovoznění BlueTooth na IBM TP T30+ s jádrem 2.4.24

Po úspěšném přeložení jádra s konfigurací: **FIXME:** popsat konfiguraci

Nejdříve přeložíme knihovnu `libbluetooth1`:

```
# apt-get source --build libbluetooth1-dev
```

Nainstalovat

```
# mv libbluetooth1*.deb /root/debs
# update-debs
# apt-get update
# apt-get install libbluetooth1-dev
```

Poté nástroje:

```
# apt-get source --build bluez-hcidump
# mv bluez-hcidump_1.5-2_i386.deb /root/debs
# update-debs
# apt-get update
# apt-get install bluez-hcidump
```

95.2. Nezpracované texty

95.2.1. Tučňák s modrým zobákem

* *Výpisky z článku jenž vyšel v CHIPu srpen 2003, strana 136 až 139*

BlueZ je oficiální implementací BT protokolu v Linuxu.

Podpora pro BT je implementována jistě v jádrech řad 2.4.x od verze 2.4.4. Pokud není přímo zakompilována je třeba přeložit jádro a tuto podporu zapnout a toto jádro nainstalovat. Výsledkem bude existence adresáře

`/lib/modules/2.4.x/kernel/net/bluetooth`. V tomto adresáři se pak budou nacházet moduly s ovladačem `bluez.o`, a moduly protokolů `l2cap.o`, `sco.o`, `bnep/bnep.o`, ...

* *Doplnit seznam modulů.*

  Do souboru `/etc/modules.conf` vložit:

```
alias net-pf-31 bluez
alias bt-proto-0 l2cap
alias bt-proto-2 sco
alias bt-proto-3 rfcomm
alias bt-proto-4 bnep
```

  Pro PC Card a jiné UART zařízení (????) ještě přidejte

```
alias tty-ldisc-15 hci-uart
```

  a pro experimenty s virtuálním HCI

```
alias char-major-10-250 hci_vhci
```

Kapitola 96. Kalendář

* ***FIXME:***

96.1. Kalendář

Kalendáře, progrogramy pro práci s kalendáři a jejich tisk.

96.1.1. cal, ncal, `gcal`

První část tvoří program **cal**, standardní kalendářový program původem z BSD Unixu a programy s ním kompatibilní.

96.1.1.1. cal

Standardní kalendářový program z BSD Unixu.

http://unicorn.us.com/cal.html

- cal
- ncal
- calendar
- xcal
- xcalendar
- pcal
- gcal

96.1.1.2. remind

The Ultimate Personal Calendar

- remind
- rem2ps
- kall
- rem
- cm2rem.tcl
- tkremind

96.1.1.3. gcal

Prints callendars

96.1.1.4. ical

http://www.research.digital.com/SRC/personal/Sanjay_Ghemawat/ical/home.html

96.1.1.5. plan

http://www.in-berlin.de/User/bitrot/plan.html

96.2. gcal

Jedná se o kalendářový program patřící ke třídě jednoduchých programů jako jsou cal, ncal, calendar

96.3. remind

Program **remind** se nachází ve stejnojmenném balíčku `remind`. Součástí balíčku jsou dále programy

rem2ps
kall
rem
cm2rem.tcl

Kapitola 97. Příprava a vypalování CD

97.1. Získání dat z audio CD, grabování

```
$ cdparanoia -B
```

97.2. cdrecord

Pro vypalování cd používám program **cdrecord**.

Pokud potřebujeme vymazat CD-RW médium, použijeme příkaz blank. Tomu zadáme jaký způsobem si přejeme mazání provést. Z manuálových stránek cdrecordu zjistíme jaké způsoby mazání jsou k dispozici, nebo se můžeme zeptat přímo programu:

```
$ cdrecord blank=help
```

Nejčastšji používám volbu mazání celého disku která se zadává `blank=all`, `blank=disc` nebo `blank=disk`.

```
$ cdrecord -v -eject blank=all
```

FIXME: -audio, -data, -mode2, -xa1, -xa2

97.3. cdrdao

Po nainstalování potřebujeme zjisti jak máme adresovat naši mechaniku. To zjistíme „skenem“ sběrnice.

```
# cdrdao scanbus
```

Mazání CD-RW média provedem příkazem **blank**

```
$ cdrdao blank --device 0,0,0 --driver generic-mmc
```

97.3.1. Instalace

Protože cdrdao se nachází v Debianu až od verze Sarge musíme mít apt nakonfigurovné aby používalo balíčky Sarge. Poté můžeme instalovat.

```
# apt-get install -t sarge cdrdao
```

Po samotné instalaci máme k dispozici několik programů:

```
-rwxr-xr-x    1 root     root       566344 Jun 24  2004 /usr/bin/cdrdao
-rwxr-xr-x    1 root     root        16916 Jun 24  2004 /usr/bin/cue2toc
-rwxr-xr-x    1 root     root       191976 Jun 24  2004 /usr/bin/toc2cddb
-rwxr-xr-x    1 root     root       195208 Jun 24  2004 /usr/bin/toc2cue
```

cdrdao je vlastní vypalovací a čtecí program, zbylé tři slouží k transformaci pomocných souborů.

Dalším krokem instalace bude zjištění jakým způsobem/adresou se dostaneme k naší vypalovací mechanice. Jako root prohledáme sběrnici

```
# cdrdao scanbus
```

97.3.2. cue2toc

Program **cdrdao** použijeme s úspěchem při vypalování obrazů, které máve ve formátu popisného `.cue` souboru s datovými soubory. V balíčku `cdrdao` se nachází program **cue2toc** který dovede `.cue` soubor převést na `.toc` soubor se kterým si **cdrdao** poradí.

```
$ cue2toc -o project.toc project.cue
$ cdrdao simulate --device 0,0,0 --driver generic-mmc project.toc
$ cdrdao write --device 0,0,0 --driver generic-mmc project.toc
```

První příkaz zkonvertuje `.cue` soubor do formátu `.toc`. Druhý příkaz je „pálení“ na sucho. Tedy takový testovací běh. Tento příkaz můžeme vypustit. Poslední příkaz proved vypálení dat popsaných v `.toc` souboru na CD.

97.4. Nezpracováno

* *condition="author"*

97.4.1. Vypalování mp3

Vypálení CD pro DVD přehravač provedu příkazem:

```
# nice --18 mkisofs -J /data/music/cd1 \
  | cdrecord -v fs=16m speed=4 dev=2,0 -
```

V první části použitím `nice --18` zvýším prioritu vypalovacímu procesu abych neriskoval přerušení vypalování v důsledku podtečení vyrovnávací paměti. Programu `mkisofs` pak přepínačem `-J` řeknu aby generoval Joliet. Používám to kvůli některým DVD přehravačům, jenž zobrazují dlouhé názvy skladeb.

```
# nice --18 mkisofs -R /data/music/mcd1 \
  | cdrecord -v fs=16m speed=4 dev=2,0 -

# mkisofs -R -o cdimage.raw /master/tree

# mount cdimage.raw -r -t iso9660 -o loop /mnt

# cdrecord -v speed=2 dev=2,0 cdimage.raw
```

97.4.2. Mazání CDRW

```
# cdrecord -v dev=2,0 blank=disc

# cdrecord -v dev=2,0 blank=fast
```

97.4.3. Postup kopírování bez chyb

Potrebuji vanilla jadro, modul cdfs.o a programy `md5sum` a `mount`

```
mkdir -p /mnt/cdrom/{cdfs,iso}
mount -o ro -t cdfs /dev/cdrom /mnt/cdrom/cdfs
X=0;
while true; do
     md5sum -b /mnt/cdrom/cdfs/* >/tmp/sumka.$X
done
while ! zblbnu; do
    diff /tmp/sumka.$a /tmp/sumka.$d
done

...
mount ...
cp /mnt/cdrom/cdfs/* /tmp/tu/
cd /tmp/tu
md5sum -b * >./sumka
cd /mnt/cdrom/cdfs
while true; do
    md5sum -c /tmp/tu/sumka
done
```

Autor doporucil CDROM i vypalovacku znacky Teac.

Kapitola 98. Udržování konfigurace

Postupy, technologie a automatizace při správě konfiguračních souborů

V této kapitole se věnuji postupům při udržování konfiguračních souborů.

98.1. Primitivní způsoby

Konfigurační soubory editujeme pomocí obyčejného textového editoru například vi, emacs či jiného. Z tohoto pravidla jsou výjimky. Pro změny v některých souborech spouštíme vyhrazené a nebo speciální konfigurační programy.

Tabulka 98-1. Doporučené způsoby úpravy některých konfiguračních souborů

soubor	program či příkaz
`/etc/passwd`	**vipw**
`/etc/group`	**vigr**
`/etc/sudoers`	**visudo**

Nejjednodušším vylepšením prosté úpravy konfiguračních souborů je udržování kopie původních verzí. Můžeme si pořídi kopii všech souborů tarem a uložit na bezpečné místo, či před editací každého souboru udělat jeho záložní kopii na místě. Záložní kopii pak pojmenujem podle původního souboru kdy za jméno přidáme příponu například `.sav`. Je to velmi jednoduché a poslouží svému účelu. První významnou nevýhodou je že máme jen jednu starší verzi a druhou že na vytváření těchto souborů musíme neustále myslet. Můžeme si ovšem pomoci skriptem podobným následujícímu.

```
#!/bin/sh
# Obálka (wrapper) kolem editace konfiguračních souborů
# WARNING: Netestováno
FNAME=$1
cp $FNAME $FNAME.new
vi $FNAME.new
cp $FNAME $FNAME.sav
mv $FNAME.new $FNAME
```

98.2. Využití RCS

Velikým krokem vpřed je použití některého systému správy verzí jako je například RCS. Umožní nám to udržovat celou historii změn v konfiguračních souborech. Pro zjednodušení si opět připravíme skript.

Příklad 98-1. `rcsvi`

```
#!/bin/sh

/usr/bin/co -l $1 && /usr/bin/vi $1 ; /usr/bin/ci -u $1

exit
```

Luxusnější verze

Příklad 98-2. `rvi`

```
#!/bin/sh

co -l $1
if [ $? -ne 0 ] ; then
        echo "Check out failed."
        echo "Maybe someone else is currently editing this file."
        echo "Aborting rvi."
        exit 1
fi
/usr/bin/rvim $1
ci -u $1
```

98.3. Systém skriptů HBIDS

Host Based Intrusion Detection System

Dalším vylepšením je obalení předchozí varianty sadou skriptů která umožní nejen udržování historie, ale také uložení této na zabezpečeném počítači a s automatickým udržováním kdy nemusíme myslet na ukládání souboru do systému správy verzí. Takovýto systém je pak použitelný i jako primitivní Systém detekce průniků (*Intrusion Detection System*)

Kontrola a udržování konfigurací počítačů (hostů) na dálku s detekcí změn. Napsáno podle „Verifying Filesystem Integrity with CVS“ v Linux Journal č. 40 February 2002

Pro sledování použijeme vyhrazený bezpečný stroj. Na tomto stroji pod uživatelem, například *hbids* budeme spouštět skripty. Přesněji necháme démona cron spouštět v pravidelných intervalech, například 15 minut, skript 98-4.

„Systém“ sledování konfiguračních a jiných souborů sestává ze dvou skriptů. První collector.pl provádí sběr dat z jednotlivých sledovaných strojů (hostů).

Příklad 98-3. `collector.pl`

```
#!/usr/bin/perl -w
# $Id: collector.pl,v 1.1.1.1 2009-01-24 15:42:52 radek Exp $

use Net::SSH::Perl;
use strict;

my %Cmds;
my $host = qw(10.16.66.19);
my $user = "root";

my @md5files = qw(/bin/login
    /usr/bin/passwd
    /bin/ps
    );

my @configfiles = qw(/etc/passwd
```

```
       /etc/shadow
       /etc/inetd.conf
       /etc/services
       );

$Cmds{'md5sigs'}     = "md5sum @md5files";
$Cmds{'configs.tar'} = "tar cf - @configfiles";
$Cmds{'suidfiles'}   = "find / -type f -perm +6000 | xargs ls -l";


### main loop ###
for my $file (keys %Cmds) {
    my $cmd = $Cmds{$file};

    # run each command on $host and print the output to $file
    &run_command($cmd, $file, $host);
}
exit 0;

sub run_command() {
    my ($cmd, $file, $host) = @_;

    # turn on compression across the ssh session
    my $ssh = Net::SSH::Perl->new($host, compression=>1);
    $ssh->login($user);
    my ($stdout, $stderr, $exit) = $ssh->cmd($cmd);

    open F, "> $file";
    print F $stdout;
    close F;
    return;
}
```

Příklad 98-4. `cvschecker.pl`

```
    #!/usr/bin/perl
    # $Id: cvschecker.pl,v 1.1.1.1 2009-01-24 15:42:52 radek Exp $

    use File::Find;
    use strict;
    use vars qw(@Files);

    my $email = 'root@localhost';
    my $host = "192.168.10.5";
    my $cvsroot = "/usr/local/hbids_cvs";
    my $collectorCmd = "/usr/local/bin/collector.pl";

    system "cvs -d $cvsroot checkout -ko $host";
    chdir $host;
    system "$collectorCmd";  # get the files from $host

    find(\&findfiles, "/home/mbr/${host}");

    # check to see if any file has been changes
    for my $file (@files) {
        if ('cvs -d $cvsroot commit -m . $file') {
```

```
 my $cvsfile = $file;
 $cvsfile =~ s|^\S+?$host|$host|;
 $cvsfile = $cvsroot ."/" . $cvsfile .",v";
 my $prerev = "";

 # use rlog to get the previous revision
 open RLOG, "rlog $cvsfile|";
 my $found = 0;
 while (<RLOG>) {
     if (/^revision\s(\S+)/) {
  $prerev = $1;
  $found++;
     }
     last if ($found == 2);
 }
 close RLOG;

 # get the changes using 'cvs diff'
 my $diff = `cvs -d $cvsroot diff -r $prerev $file`;
 open TMP, "> /tmp/change";
 print TMP $diff;
 close TMP;
 $file =~ s|^\S+?$host||g;

 # send an mail that contains the changes
 `mail -s "Changed file on $host: $file" $email \
                 < /tmp/change`;
    }
}
exit 0;

# find all files in the local directory structure,
# but ignore directories and CVS files
sub findfiles() {
    my $file = $File::Find::name;
    unless (-d $file || $file =~ /CVS/) {
 push @Files, $file;
    }
    return;
}                                                          ❶
```

❶ Tento skript je napsán v Perlu.

98.4. Databáze konfigurací s historií a detekcí změn

Příklad systému.

Systém se skládá z řady spolupracujících skriptů běžících na vyhrazeném zabezpečeném počítači. Princip jeho funkce je následující. Sledovací počítač má pro každý sledovaný stroj databázi souborů jež sleduje. V pravidelných interval jen pak pomocí programu cron spouštěn hlavní skript jenž provádí sběr údajů, jejich porovnávání s předcházejícím stavem a ukládání změn do systému správy verzí. Další modul je pak zodpovědný za hlášení změn správci, či správcům jednotlivých sledovaných počítačů například pomocí elektronické pošty a/nebo SMS správa

na mobilní telefon či pager. Dalším rozšířením pak může být sledování kontrolních součtů binárních souborů programů.

Kapitola 99. Palm Pilot

Propojení stanice s Palm Pilotem

* *chapter id="palm-pilot"*

Seznam balíčků v Debian GNU/Linux

- coldsync
- libpalm-perl
- jpilot
- linpqa
- pilot-manager
- pilot-template
- pilrc
- prc-tools
- gnome-pilot
- pilot-link
- plucker
- kpilot
- pilot-link-perl
- autopilot

Zálohu dat v Palmovi vytvoříme docela snadno pomocí programu pilot-xfer z balíčku pilot-link.

Příklad 99-1. Záloha dat z Palma na WS

```
$ pilot-xfer -s adresář
```

Uvedený příklad nám vytvoří zálohu do daného adresáře a byla-li již vytvořena tak ji synchronizuje (ozálohuje jen co bylo změněno).

Instalace programů je také snadná.

Příklad 99-2. Instalace programu do Palma

```
$ pilot-xfer -i prc_file
```

stejně jako instalace poznámek (memo)

Příklad 99-3. Instlace poznámek do Palma

```
$ install-meme -c kategorie memo_file
```

99.1. Balíček programů `pilot-link`

Základní nástroje pro komunikaci s Palm Pilotem

Seznam programů v balíčku:

- pilot-xfer
- install-memo
- memos - přečte všechny či jen vybrané poznámky buď přímo z palma, nebo z databáze MemoDB.pdb

99.1.1. Instalace

* *V Debian GNU/Linux jsou k dispozici k 2003-09-17 tyto verze `0.9.5.0-8` (woody, sarge), `0.11.8-5` (sid).*

První pokus o připojení Palma M515

```
Sep 17 11:58:44 yoda kernel: hub.c: port 1, portstatus 101, change 1, 12 Mb/s
Sep 17 11:58:44 yoda kernel: hub.c: port 1 connection change
Sep 17 11:58:44 yoda kernel: hub.c: port 1, portstatus 101, change 1, 12 Mb/s
Sep 17 11:58:44 yoda kernel: hub.c: port 1, portstatus 101, change 0, 12 Mb/s
Sep 17 11:58:45 yoda last message repeated 3 times
Sep 17 11:58:45 yoda kernel: hub.c: port 1, portstatus 103, change 0, 12 Mb/s
Sep 17 11:58:45 yoda kernel: hub.c: new USB device 00:1d.1-1, assigned address 8
Sep 17 11:58:45 yoda kernel: usb.c: kmalloc IF c6b81f40, numif 1
Sep 17 11:58:45 yoda kernel: usb.c: new device strings: Mfr=1, Product=2, SerialNumber=5
Sep 17 11:58:45 yoda kernel: usb.c: USB device number 8 default language ID 0x409
Sep 17 11:58:45 yoda kernel: Manufacturer: Palm, Inc.
Sep 17 11:58:45 yoda kernel: Product: Palm Handheld
Sep 17 11:58:45 yoda kernel: SerialNumber: L0RP15F21198
Sep 17 11:58:45 yoda kernel: usb.c: unhandled interfaces on device
Sep 17 11:58:45 yoda kernel: usb.c: USB device 8 (vend/prod 0x830/0x3) is not claimed by any active driver.
Sep 17 11:58:45 yoda kernel:   Length              = 18
Sep 17 11:58:45 yoda kernel:   DescriptorType      = 01
Sep 17 11:58:45 yoda kernel:   USB version         = 1.00
Sep 17 11:58:45 yoda kernel:   Vendor:Product      = 0830:0003
Sep 17 11:58:45 yoda kernel:   MaxPacketSize0      = 16
Sep 17 11:58:45 yoda kernel:   NumConfigurations   = 1
Sep 17 11:58:45 yoda kernel:   Device version      = 1.00
Sep 17 11:58:45 yoda kernel:   Device Class:SubClass:Protocol = 00:00:00
Sep 17 11:58:45 yoda kernel:     Per-interface classes
Sep 17 11:58:45 yoda kernel: Configuration:
Sep 17 11:58:45 yoda kernel:   bLength             =    9
Sep 17 11:58:45 yoda kernel:   bDescriptorType     =   02
Sep 17 11:58:45 yoda kernel:   wTotalLength        = 002e
Sep 17 11:58:45 yoda kernel:   bNumInterfaces      =   01
Sep 17 11:58:45 yoda kernel:   bConfigurationValue =   01
Sep 17 11:58:45 yoda kernel:   iConfiguration      =   00
Sep 17 11:58:45 yoda kernel:   bmAttributes        =   c0
Sep 17 11:58:45 yoda kernel:   MaxPower            =    2mA
Sep 17 11:58:45 yoda kernel:
Sep 17 11:58:45 yoda kernel:   Interface: 0
Sep 17 11:58:45 yoda kernel:   Alternate Setting:  0
Sep 17 11:58:45 yoda kernel:     bLength             =    9
Sep 17 11:58:45 yoda kernel:     bDescriptorType     =   04
Sep 17 11:58:45 yoda kernel:     bInterfaceNumber    =   00
Sep 17 11:58:45 yoda kernel:     bAlternateSetting   =   00
Sep 17 11:58:45 yoda kernel:     bNumEndpoints       =   04
Sep 17 11:58:45 yoda kernel:     bInterface Class:SubClass:Protocol =   ff:00:00
Sep 17 11:58:45 yoda kernel:     iInterface          =   00
Sep 17 11:58:45 yoda kernel:     Endpoint:
Sep 17 11:58:45 yoda kernel:       bLength             =    7
Sep 17 11:58:45 yoda kernel:       bDescriptorType     =   05
Sep 17 11:58:45 yoda kernel:       bEndpointAddress    =   81 (in)
Sep 17 11:58:45 yoda kernel:       bmAttributes        =   02 (Bulk)
Sep 17 11:58:45 yoda kernel:       wMaxPacketSize      = 0010
Sep 17 11:58:45 yoda kernel:       bInterval           =   00
Sep 17 11:58:45 yoda kernel:     Endpoint:
Sep 17 11:58:45 yoda kernel:       bLength             =    7
Sep 17 11:58:45 yoda kernel:       bDescriptorType     =   05
Sep 17 11:58:45 yoda kernel:       bEndpointAddress    =   01 (out)
Sep 17 11:58:45 yoda kernel:       bmAttributes        =   02 (Bulk)
Sep 17 11:58:45 yoda kernel:       wMaxPacketSize      = 0010
Sep 17 11:58:45 yoda kernel:       bInterval           =   00
Sep 17 11:58:45 yoda kernel:     Endpoint:
Sep 17 11:58:45 yoda kernel:       bLength             =    7
Sep 17 11:58:45 yoda kernel:       bDescriptorType     =   05
Sep 17 11:58:45 yoda kernel:       bEndpointAddress    =   82 (in)
Sep 17 11:58:45 yoda kernel:       bmAttributes        =   02 (Bulk)
Sep 17 11:58:45 yoda kernel:       wMaxPacketSize      = 0040
Sep 17 11:58:45 yoda kernel:       bInterval           =   00
Sep 17 11:58:45 yoda kernel:     Endpoint:
Sep 17 11:58:45 yoda kernel:       bLength             =    7
Sep 17 11:58:45 yoda kernel:       bDescriptorType     =   05
Sep 17 11:58:45 yoda kernel:       bEndpointAddress    =   02 (out)
Sep 17 11:58:45 yoda kernel:       bmAttributes        =   02 (Bulk)
Sep 17 11:58:45 yoda kernel:       wMaxPacketSize      = 0040
Sep 17 11:58:45 yoda kernel:       bInterval           =   00
Sep 17 11:58:45 yoda kernel: usb.c: kusbd: /sbin/hotplug add 8
Sep 17 11:58:45 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 11:58:45 yoda kernel: hub.c: port 1, portstatus 103, change 0, 12 Mb/s
Sep 17 11:58:45 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 11:58:45 yoda /sbin/hotplug: arguments (usb) env (PWD=/etc/hotplug DEVICE=/proc/bus/usb/002/008 INTERFACE=255/0/0 ACTION=add
DEBUG=kernel OLDPWD=/ DEVFS=/proc/bus/usb TYPE=0/0/0 SHLVL=1 HOME=/ PATH=/bin:/sbin:/usr/sbin:/usr/bin PRODUCT=830/3/100 _=/usr/bin/
env)
Sep 17 11:58:45 yoda /sbin/hotplug: invoke /etc/hotplug/usb.agent ()
Sep 17 11:58:45 yoda /etc/hotplug/usb.agent: Setup visor for USB product 830/3/100
Sep 17 11:58:45 yoda kernel: usb.c: registered new driver serial
Sep 17 11:58:45 yoda kernel: usbserial.c: USB Serial support registered for Generic
Sep 17 11:58:45 yoda kernel: usbserial.c: Generic converter detected
Sep 17 11:58:45 yoda kernel: usbserial.c: Generic converter now attached to ttyUSB0 (or usb/tts/0 for devfs)
Sep 17 11:58:45 yoda kernel: usbserial.c: Generic converter now attached to ttyUSB1 (or usb/tts/1 for devfs)
Sep 17 11:58:45 yoda kernel: usb.c: serial driver claimed interface c6b81f40
Sep 17 11:58:45 yoda kernel: usbserial.c: USB Serial Driver core v1.4
Sep 17 11:58:45 yoda kernel: usbserial.c: USB Serial support registered for Handspring Visor / Palm 4.0 / Clié 4.x
Sep 17 11:58:45 yoda kernel: usbserial.c: USB Serial support registered for Sony Clié 3.5
Sep 17 11:58:45 yoda kernel: visor.c: USB HandSpring Visor, Palm m50x, Sony Clié driver v1.6
Sep 17 11:58:45 yoda /etc/hotplug/usb.agent: missing kernel or user mode driver visor
Sep 17 11:59:55 yoda kernel: hub.c: port 1, portstatus 100, change 3, 12 Mb/s
Sep 17 11:59:55 yoda kernel: hub.c: port 1 connection change
Sep 17 11:59:55 yoda kernel: hub.c: port 1, portstatus 100, change 3, 12 Mb/s
Sep 17 11:59:55 yoda kernel: usb.c: USB disconnect on device 00:1d.1-1 address 8
Sep 17 11:59:55 yoda kernel: usbserial.c: Generic converter now disconnected from ttyUSB0
```

```
Sep 17 11:59:55 yoda kernel: usbserial.c: Generic converter now disconnected from ttyUSB1
Sep 17 11:59:55 yoda kernel: usb.c: kusbd: /sbin/hotplug remove 8
Sep 17 11:59:55 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 11:59:55 yoda /sbin/hotplug: arguments (usb) env (PWD=/etc/hotplug DEVICE=/proc/bus/usb/002/008 INTERFACE=255/0/0 ACTION=remo
ve DEBUG=kernel OLDPWD=/ DEVFS=/proc/bus/usb TYPE=0/0/0 SHLVL=1 HOME=/ PATH=/bin:/sbin:/usr/sbin:/usr/bin PRODUCT=830/3/100 _=/usr/b
in/env)
Sep 17 11:59:55 yoda kernel: hub.c: port 1, portstatus 100, change 2, 12 Mb/s
Sep 17 11:59:55 yoda kernel: hub.c: port 1 enable change, status 100
Sep 17 11:59:55 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 11:59:55 yoda /sbin/hotplug: invoke /etc/hotplug/usb.agent ()
```

```
# modprobe usbserial
```

```
Sep 17 12:14:02 yoda kernel: hub.c: port 1, portstatus 101, change 1, 12 Mb/s
Sep 17 12:14:02 yoda kernel: hub.c: port 1 connection change
Sep 17 12:14:02 yoda kernel: hub.c: port 1, portstatus 101, change 1, 12 Mb/s
Sep 17 12:14:02 yoda kernel: hub.c: port 1, portstatus 101, change 0, 12 Mb/s
Sep 17 12:14:02 yoda last message repeated 3 times
Sep 17 12:14:02 yoda kernel: hub.c: port 1, portstatus 103, change 0, 12 Mb/s
Sep 17 12:14:02 yoda kernel: hub.c: new USB device 00:1d.1-1, assigned address 9
Sep 17 12:14:02 yoda kernel: usb.c: kmalloc IF d568d400, numif 1
Sep 17 12:14:02 yoda kernel: usb.c: new device strings: Mfr=1, Product=2, SerialNumber=5
Sep 17 12:14:02 yoda kernel: usb.c: USB device number 9 default language ID 0x409
Sep 17 12:14:02 yoda kernel: Manufacturer: Palm, Inc.
Sep 17 12:14:02 yoda kernel: Product: Palm Handheld
Sep 17 12:14:02 yoda kernel: SerialNumber: L0RP15F21198
Sep 17 12:14:02 yoda kernel: usbserial.c: Handspring Visor / Palm 4.0 / Clié 4.x converter detected
Sep 17 12:14:03 yoda kernel: visor.c: Handspring Visor / Palm 4.0 / Clié 4.x: Number of ports: 2
Sep 17 12:14:03 yoda kernel: visor.c: Handspring Visor / Palm 4.0 / Clié 4.x: port 1, is for Generic use and is bound to ttyUSB0
Sep 17 12:14:03 yoda kernel: visor.c: Handspring Visor / Palm 4.0 / Clié 4.x: port 2, is for HotSync use and is bound to ttyUSB1
Sep 17 12:14:03 yoda kernel: usbserial.c: Handspring Visor / Palm 4.0 / Clié 4.x converter now attached to ttyUSB0 (or usb/tts/0 for devfs)
Sep 17 12:14:03 yoda kernel: usbserial.c: Handspring Visor / Palm 4.0 / Clié 4.x converter now attached to ttyUSB1 (or usb/tts/1 for devfs)
Sep 17 12:14:03 yoda kernel: usb.c: serial driver claimed interface d568d400
Sep 17 12:14:03 yoda kernel: usb.c: kusbd: /sbin/hotplug add 9
Sep 17 12:14:03 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 12:14:03 yoda kernel: hub.c: port 1, portstatus 103, change 0, 12 Mb/s
Sep 17 12:14:03 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 12:14:03 yoda /sbin/hotplug: arguments (usb) env (PWD=/etc/hotplug DEVICE=/proc/bus/usb/002/009 INTERFACE=255/0/0 ACTION=add DEBUG=kernel OLDPWD=/ DEVFS=/proc/bus/usb
Sep 17 12:14:03 yoda /sbin/hotplug: invoke /etc/hotplug/usb.agent ()
Sep 17 12:14:03 yoda /etc/hotplug/usb.agent: Setup visor for USB product 830/3/100
Sep 17 12:14:03 yoda /etc/hotplug/usb.agent: missing kernel or user mode driver visor
```

```
# modprobe visor
```

```
Sep 17 12:17:28 yoda kernel: hub.c: port 1, portstatus 101, change 1, 12 Mb/s
Sep 17 12:17:28 yoda kernel: hub.c: port 1 connection change
Sep 17 12:17:28 yoda kernel: hub.c: port 1, portstatus 101, change 1, 12 Mb/s
Sep 17 12:17:28 yoda kernel: hub.c: port 1, portstatus 101, change 0, 12 Mb/s
Sep 17 12:17:28 yoda last message repeated 3 times
Sep 17 12:17:29 yoda kernel: hub.c: port 1, portstatus 103, change 0, 12 Mb/s
Sep 17 12:17:29 yoda kernel: hub.c: new USB device 00:1d.1-1, assigned address 10
Sep 17 12:17:29 yoda kernel: usb.c: kmalloc IF c6b81e40, numif 1
Sep 17 12:17:29 yoda kernel: usb.c: new device strings: Mfr=1, Product=2, SerialNumber=5
Sep 17 12:17:29 yoda kernel: usb.c: USB device number 10 default language ID 0x409
Sep 17 12:17:29 yoda kernel: Manufacturer: Palm, Inc.
Sep 17 12:17:29 yoda kernel: Product: Palm Handheld
Sep 17 12:17:29 yoda kernel: SerialNumber: L0RP15F21198
Sep 17 12:17:29 yoda kernel: usbserial.c: Handspring Visor / Palm 4.0 / Clié 4.x converter detected
Sep 17 12:17:29 yoda kernel: visor.c: Handspring Visor / Palm 4.0 / Clié 4.x: Number of ports: 2
Sep 17 12:17:29 yoda kernel: visor.c: Handspring Visor / Palm 4.0 / Clié 4.x: port 1, is for Generic use and is bound to ttyUSB0
Sep 17 12:17:29 yoda kernel: visor.c: Handspring Visor / Palm 4.0 / Clié 4.x: port 2, is for HotSync use and is bound to ttyUSB1
Sep 17 12:17:29 yoda kernel: usbserial.c: Handspring Visor / Palm 4.0 / Clié 4.x converter now attached to ttyUSB0 (or usb/tts/0 for devfs)
Sep 17 12:17:29 yoda kernel: usbserial.c: Handspring Visor / Palm 4.0 / Clié 4.x converter now attached to ttyUSB1 (or usb/tts/1 for devfs)
Sep 17 12:17:29 yoda kernel: usb.c: serial driver claimed interface c6b81e40
Sep 17 12:17:29 yoda kernel: usb.c: kusbd: /sbin/hotplug add 10
Sep 17 12:17:29 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 12:17:29 yoda kernel: hub.c: port 1, portstatus 103, change 0, 12 Mb/s
Sep 17 12:17:29 yoda kernel: hub.c: port 2, portstatus 100, change 0, 12 Mb/s
Sep 17 12:17:29 yoda /sbin/hotplug: arguments (usb) env (PWD=/etc/hotplug DEVICE=/proc/bus/usb/002/010 INTERFACE=255/0/0 ACTION=add DEBUG=kernel OLDPWD=/ DEVFS=/proc/bus/usb
Sep 17 12:17:29 yoda /sbin/hotplug: invoke /etc/hotplug/usb.agent ()
Sep 17 12:17:29 yoda /etc/hotplug/usb.agent: Setup visor for USB product 830/3/100
Sep 17 12:17:29 yoda /etc/hotplug/usb.agent: missing kernel or user mode driver visor
```

```
# apt-get install pilot-link
```

99.1.2. memos

Příklad 99-4. Čtení „poznámek“ z projektu RISC8

```
$ memos -t RISC8 -d .
```

99.2. Nezařazené texty

99.2.1. Připojení Palm M515 přes USB

```
# insmod usbserial vendor=0x830 product=0x3
# pilot-link -p /dev/usb/tts/1 -l
```

V deníku by se mělo objevit něco jako

```
Nov 10 15:55:10 matilda /sbin/hotplug: arguments (usb) env (DEVFS=/proc/bus/usb OLDPWD=/ PATH=
Nov 10 15:55:10 matilda /sbin/hotplug: invoke /etc/hotplug/usb.agent ()
Nov 10 15:55:13 matilda /etc/hotplug/usb.agent: Setup visor for USB product 830/3/100
```

99.2.2. ColdSync

Dalším programem prostřednictvím kterého můžeme synchronizovat svůj Palm je program ColdSync (http://www.coldsync.org/)

Kapitola 100. Sazba

* *chapter id="typesetting" condition="author"*

Odkazy a zdroje:

- netpbm (http://netpbm.sourceforge.net/doc/)
- Troff Resources (http://www.kohala.com/start/troff/troff.html)

100.1. groff/troff/nroff

FIXME:

Jedná se snad o historicky nejstarší formátovací a sázecí systém který znám. Vychází z programu **runoff** jenž prováděl sazbu textu jen pro řádkové tiskárny (znaky ze stejnou šířkou, monospace). Tento byl pod názvem nroff portován na první UNIX a posléze rozšířen v programu troff pro sazbu textů se znaky s proměnnou/různou šířkou.

K samotnému sázecímu programu máme k dispozici řadu dalších jenž realizují některé části formátování a sazby pro speciální účely a nebo pomáhalí s udržováním rejstříků a indexů.

- **groff** — front-end for the groff document formatting system
- **troff** —
- **eqn** —
- **grn** —
- **pic** —
- **refer** —
- **soelim** —
- **tbl** — format tables for troff
- **grap** —

100.1.1. Knihovny maker

Pro **groff** existuje několik knihoven maker.

Tabulka 100-1. Knihovna maker pro groff

označení	popis
man	referenční příručky, dokumentace počítačů
ms	vhodné pro zprávy, dopisy, knihy, uživatelské manuály apod.
mm	úřední dokumentace. mm makra jsou kompatibilní s původími makry v DWB
me	časopisecké články, konferenční příspěvky
mdoc	
mom	

100.1.1.1. ms

Makra pro titulní stránku

.RP [no]

Specifukuje formát dokumentu.

.P1

FIXME:

.DA

FIXME:

.ND

FIXME:

.TL

Titulek dokumentu.

.AU

Jméno autora. Je možno zadat více autorů.

.AI

Instituce autora. Je možno specifikovat více institucí.

.AB [no]

Začátek abstraktu. Volba no potlačuje tisk slova „ABSTRACT“ nad vlastním textem abstraktu.

.AE

Konec abstraktu.

100.1.2. Použití groffu k vytváření obrázků

Odkazy a zdroje:

- Turning PIC into HTML (http://www.kohala.com/start/troff/pic2html.html)

Systém automatické sazby **groff** se dá použít i pro vytváření samostatných obrázků.

$

100.2. TeX

FIXME:

Kapitola 101. Přehled distribucí Linuxu

* *chapter id="distribuce"*

Odkazy a zdroje:

- DistroWatch (http://www.distrowatch.com)
- Knoppix Customizations (http://www.knoppix.net/docs/index.php/KnoppixCustomizations)

101.1. Malé distribuce

Distribuce na jednu či více disket a na jedno CD.

LNXBBC (http://www.lnx-bbc.org/) — Linux Bootable Business Card

Distribuce na CD velikost 50MB. Tzv. vizitka. To je CD tvaru a velikosti vizitky.

Crash Recovery Kit for Linux (http://www.crashrecovery.org/)

FIXME:

Trinux (http://trinux.sourceforge.net/)

FIXME:

Superrescue (ftp://ftp.kernel.org/pub/dist/superrescue/)

FIXME:

Eisfair (http://www.eisfair.org/)

FIXME:

fli4l (http://www.fli4l.de/)

FIXME:

hal91 (http://jspiro.tripod.com/linux/hal91.htm)

FIXME:

Linux - Small Kernel Project (http://www.superant.com/smalllinux/)

Pro 386 s 2MB RAM

VectorLinux (http://ibiblio.org/vectorlinux/)

FIXME:

Distribuce na diskety

tomsrtbt (http://www.toms.net/rb/)

Jednodisketový linux. "The most GNU/Linux on 1 floppy disk."

Distribuce na Business Card

Damn Small Linux (http://www.damnsmalllinux.org/)

FIXME:

Pilot Linux (http://www.pilotlinux.nl/)

Jednoúčelový linux jako klient k MS terminal serveru. Je založen na Damn Small Linuxu.

czfdebian (http://www.czfree.net/czfdebian/)

Malá distribuce určená pro routery CZFree. Je určena pro CF karty a založena na Debianu jak již název napovídá.

Distribuce na mini CD

MiniCD (http://minicd.berlios.de/)

Jedno miniCD live system založený na Mandrake

101.2. Záchranné CD

System Rescue CD-Rom (http://www.sysresccd.org/)

Založeno na distribuci Gentoo.

Kapitola 102. LEAF

Distribuce pro směrovače a firewally

102.1. Instalace

Nejdříve si musíme zformátovat disketu na 1.68MB. K tomu použijme program `format` nebo `superformat`

```
# superformat /dev/fd0u1680
```

nebo pomalejší

```
# superformat --superverify /dev/fd0u1680
```

nebo

```
# fdformat /dev/fd0u1680
```

Na takto připravenou disketu pak přeneseme obraz.

```
# dd if=Bering_1.0-rc3_img_bering_1680.bin of=/dev/fd0u1680
```

Na takto vytvořené diketě jsou soubory/balíčky:

- `bridge.lrp`
- `dhcpd.lrp`
- `dnscache.lrp`
- `etc.lrp`
- `initrd.lrp`
- `keyboard.lrp`
- `local.lrp`
- `log.lrp`
- `modules.lrp`
- `ppp.lrp`
- `pppoe.lrp`
- `pump.lrp`
- `root.lrp`
- `shorwall.lrp`
- `tc.lrp`
- `weblet.lrp`
- `ldlinux.sys`
- `linux`
- `readme`
- `syslinux.cfg`
- `syslinux.dpy`

102.2. Úpravy na obrazu diskety

102.2.1. Připojení a odpojení obrazu diskety

Nejdříve si zavedeme modul zařízení `loop`

```
# modprobe loop
```

nyní si již můžeme připojit obraz diskety

```
# losetup /dev/loop1 img.bin
# mnt /dev/loop1 /mnt/floopy
```

Po provední všech nezbytných úprav, obraz odpojíme

```
umount /dev/loop1
```

a uvolníme zařízení `loop1`

```
losetup -d /dev/loop1
```

102.2.2. Úprava modulů

Z připojeného obrazu diskety si do podadresáře rozbalíme soubor `modules.lrp`

```
# mkdir root
# cd root
# tar xzvf /mnt/floppy/modules.lrp
```

Do rozbalených adresářů a souborů, do adresáře `lib/modules` přidáme potřebné moduly získané ze souboru `Bering_1.0-rc3_modules_2.4.18.tar.gz`. V mém případě jsem si přidal modul `3c509.o` a odstranil `ne2k-pci.o`, `n_hdlc.o` a všechny moduly kolem `ppp`

Poté je třeba upravit soubor `etc/modules` kde jsou uvedeny moduly jenž se mají při strtu systému zavést do paměti. V mém souboru zůstalo

```
3c509
8390
ne io=0x240

ip_conntrack_ftp
ip_conntrack_irc
ip_nat_ftp
ip_nat_irc
```

Po úpravách vytvoříme balíček

```
tar cf - *| gzip -9 >/mnt/floppy/modules.lrp
```

102.3. Poznámky ke konfiguraci

Pár poznámek ke konfiguračním soborům.

102.3.1. Editor

V LEAF Bering je jako editor použit `e3`. Tento se nachází v adresáři `/bin` včetně symbolických odkazů>

e3em
e3ne
e3pi
e3vi
e3ws

Jak je patrno s odkazů, `e3` umí emulovat chování jiných editorů.

V adresáři `/bin` jsou ještě dva soubory

- `/bin/editor` - jnenž je symbolickým odkazem na některý z e3em, e3ne, e3pi, e3vi, e3ws
- `/bin/edit` - skript pro spouštění editoru

102.3.2. Balíčky které se nezavádějí

Balíčky které se zavedou jsou uvedeny v souboru `/syslinux.cfg` v obrazu diskety. Jsou uvedeny v parametru `LRP`

Kapitola 103. Wiki Wiki

Wiki Wiki Web

* *chapter id="wiki" xreflabel="Wiki Wiki"*

Odkazy a zdroje:

- Wiki (http://wiki.org/)
- What Is Wiki (http://wiki.org/wiki.cgi?WhatIsWiki)
- The Kwiki Home Page (http://www.kwiki.org/)
- TWiki™ - an Enterprise Collaboration Platform (http://www.twiki.org/)
- Media Wiki (http://wikipedia.sourceforge.net/)
- Apache::MiniWiki (http://www.nyetwork.org/wiki/MiniWiki)
- Puki Wiki (http://pukiwiki.org/?About%20PukiWiki)
- Swiki Swiki na platforme Squeak (http://minnow.cc.gatech.edu/swiki)

Co je to wiki? Wiki je platforma pro spolupráci, wiki je veřejná tabule. Místo které smí každý číst, každý psát a měnit. Tyhle slova jsou to nejvýstižnější co mne napadlo. Chcete-li, můžete si představit wiki web jako web, kde je možno každou stránku z prohlížeče měnit. Připisovat další text, mazat. Ačkoli to zní na první pohled značně zvláštně, princip wiki se osvědčil. Za jedeny z největších úspěchů můžeme směle označit.

- Portland Pattern Repository's Wiki (http://c2.com/cgi/wiki?WelcomeVisitors)
- Wikipedia (http://wikipedia.org/)

103.1. MiniWiki

MiniWiki je velmi jednoduchý wiki program psaný v Perlu běžící pod Apache.

103.2. TWiki

* *section id="twiki" xreflabel="TWiki"*

Odkazy:

- TWiki home page (http://www.twiki.org), Collaborate with TWiki.
- http://www.twiki.org

TWiki je jedním z prvních systémů který jsem zkoušel nasadit. Hlavním důvodem bylo že jedny z prvních wiki webů které jsem používal či aktivně sledoval byly postaveny právě na TWiki. To co na TWiki hodnotím nejvíce je:

- Text stránek uložen v obyčejných textových souborech, tedy snadno zpracovatelném formátu.
- Historie, všechny verze a změny jsou v RCS
- Poměrně rozsáhle konfigurační možnosti.
- Možnost upozornění na změny mailem.
- Poměrně podrobné řízení přístupu uživatelů.

Uvedené vlastnosti se mi jeví jako velmi vhodné pro nasazení v korporátním prostředí i pro vytváření komunitních webů jež nejsou zcela veřejné.

103.2.1. Instalace

103.2.1.1. Instalace TWiki 4.0.1

Stáhneme si balíček s verzí. Na serveru si vytvoříme adresář `/usr/local/share/twiki` a do tohoto adresáře balíček rozbalíme.

> **Varování**
>
> Při rozbalování se nevytvoří adresář, ale všechno se rozbalí na místě. Adresář si musíme vytvořit sami a rozbalovat v něm.

Nainstalujeme si potřebný software:

```
# aptitude install apache2 libcgi-perl libdigest-sha1-perl libcgi-session-perl rcs patch libne
```

103.2.1.2. Instalace TWiki 4.0.5

Stáhneme si balíček s verzí. Na serveru si vytvoříme adresář `/usr/local/share/twiki` a do tohoto adresáře balíček rozbalíme.

> **Varování**
>
> Při rozbalování se nevytvoří adresář, ale všechno se rozbalí na místě. Adresář si musíme vytvořit sami a rozbalovat v něm.

```
# mkdir -p /usr/local/download/twiki
# cd /usr/local/download/twiki
# wget http://twiki.org/p/pub/Codev/Release/TWiki-4.0.5.tgz
# mkdir -p /usr/local/share/twiki
# cd /usr/local/share/twiki
# tar xzvf /usr/local/download/twiki/TWiki-4.0.5.tgz
```

Nainstalujeme si potřebný software:

```
# aptitude install apache2 libcgi-perl libdigest-sha1-perl libcgi-session-perl rcs patch libne

# chown -R www-data:www-data /usr/local/share/twiki
# chmod -R u+rw /usr/local/share/twiki
# cd /usr/local/share/twiki/bin
# cp LocalLib.cfg.txt LocalLib.cfg
# vi LocalLib.cfg

# TWiki Apache configuration
ScriptAlias /twiki/bin /usr/local/share/twiki/bin
Alias /twiki /usr/local/share/twiki
<Directory /usr/local/share/twiki>

</Directory>

<Directory /usr/local/share/twiki/bin>
        Options ExecCGI
        SetHandler cgi-script
        AllowOverride All
        Allow From All
```

```
</Directory>
```

103.3. Nezapracované poznámky

Přehled Wiki

Kandidáti:
aswiki ruby1.6,
didiwiki velmi malé
kwiki jen perl, CGI
twiki Woody Backports

aswiki
Psáno v Ruby, je v Sarge. Používá ruby1.6.
 * use template library Amrita, so we can customize its output HTML easyly.
 * RCS base version log.
 * plugin system.
 * file attach.

hiki
Psáno v Ruby, je v Sarge. Používá ruby1.8

didiwiki
Psáno v C, je v Sarge. Obsahuje webserver, velim malá binárka.

kwiki

twiki
ukládá data do RCS, Debian Sarge, Woody Backports.

moinmoin
v Pythonu

Kapitola 104. Groupware WORKING

Programy pro týmovou práci

* *chapter id="groupware"*

V této kapitole se zabývám nástroji pro podporu práce v týmu. Je to programové vybavení třídy groupware. Není čas ani příležitost zkoušet všechno. Proto si nejdříve posbírám inforamce k čemu a za jakých okolností se bude systém používat, a potom pečlivě prozkoumám dostupné systémy. Teprve pak vyberu nějaký ke skutečnému odzkoušení. Proto zde není popsáno mnoho groupware systémů.

104.1. Kolab Groupware

Odkazy a zdroje:

- Kolab Groupware Project (http://www.kolab.org/)

Na systém Kolab jsem narazil když jsem hledal řešení jenž integruje program MS Outlook. Připomínám že do MS Outlooku je třeba zakoupit komerční konektor. Buď'to Toltec Connector nebo Konsec Konnektor. Je možné že se čase objeví i jiné, případně i open source konektory.

104.1.1. Ukázka instalace

Instalace na virtuálním serveru `sol` do virtuálního stroje `kol2`. Jako firemní doména je v ukázce `example.cz`.

```
sol:~# rebuild-vserver kol2 10.16.66.140
```

- Time zone configuration: **`Europe/Prague`**
- Enable shadow passwords: **`Yes`**
- Root password: **`******`**
- Create a normal user account now: **`No`**
- Choose software to install: *nevybráno nic*

- `locales` — nakonfigurováno en-us cs-cz, nastaveno none

Stáhl jsem obsah adresáře `http://ftp.belnet.be/packages/kolab/server/release/kolab-server-2.0.3/ix86-d` do pracovního adresáře. V tomto adresáři jsem spustil příkaz:

```
# ./obmtool kolab 2>&1 | tee kolab-build.log
MISSPKG: none
MISSING: none
SURPLUS: none
SUMMARY: NODE=kol2; CMD=kolab; DATE=2006-03-01/05:41:33; HASX11=; DONE

Adding symbolic link to /kolab/bin/kolab as /usr/bin/kolab

kol2:/# /etc/init.d/kolab stop
OpenPKG: stop: spamassassin, amavisd, apache, clamav, imapd, postfix, proftpd
OpenPKG: stop: kolabd, sasl, openldap.

kol2:/# /kolab/etc/kolab/kolab_bootstrap -b
...
```

```
Please enter Hostname including Domain Name (e.g. thishost.domain.tld) [kol2]: kol2.example.cz
Proceeding with Hostname kol2.example.cz
Do you want to set up (1) a master Kolab server or (2) a slave [1] (1/2): 1
Please enter your Maildomain - if you do not know your mail domain use the fqdn from above [ex
proceeding with Maildomain example.cz
Kolab primary email addresses will be of the type user@example.cz
Generating default configuration:
Top level DN for Kolab [dc=example,dc=cz]: dc=kol2,dc=example,dc=cz
 base_dn : dc=kol2,dc=example,dc=cz
 bind_dn : cn=manager,cn=internal,dc=kol2,dc=example,dc=cz
Please choose a manager password [E+8Vrz98AsD1pwHN]: ********
...
Do you want to create CA and certificates [y] (y/n): y
Now we need to create a cerificate authority (CA) for Kolab and a server
certificate. You will be prompted for a passphrase for the CA.
################################################################################
/kolab/etc/kolab/kolab_ca.sh -newca kol2.example.cz
Enter organization name [Kolab]: : FIRMA example.cz
Enter organizational unit [Test-CA]: Kolab
Using subject O=FIRMA ,OU=Kolab,CN=kol2.example.cz
Using dn
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
.....................++++++
.................................................++++++
writing new private key to '/kolab/etc/kolab/ca/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
phrase is too short, needs to be at least 4 chars
Enter PEM pass phrase: ********
CA and certificate creation complete.

You can install /kolab/etc/kolab/ca/cacert.pem on your clients to allow them
to verify the validity of your server certificates.

kolab is now ready to run!
please run '/kolab/bin/openpkg rc all start'
Use login=manager and passwd=******** when you log into
the webinterface https://kol2.example.cz/admin !
kol2:/#

# /etc/init.d/kolab stop
OpenPKG: stop: spamassassin, amavisd, apache, clamav, imapd, postfix, proftpd
OpenPKG: stop: kolabd, sasl, openldap.
# /etc/init.d/kolab start
OpenPKG: start: openpkg, openldap, sasl, spamassassin, amavisd:FAILED
openpkg:rc:WARNING: /kolab:amavisd:%start: failed with return code 25
openpkg:rc:NOTICE: output from stdout/stderr is following:
+-----------------------------------------------------------------
|   The value of variable $myhostname is "kol2", but should have been
|   a fully qualified domain name; perhaps uname(3) did not provide such.
|   You must explicitly assign a FQDN of this host to variable $myhostname
|   in amavisd.conf, or fix what uname(3) provides as a host's network name!
+-----------------------------------------------------------------
OpenPKG: start: apache, clamav, imapd, postfix, proftpd, kolabd.
```

104.2. Groupware a MS Windows

Protože vybraný groupware potřebují používat i uživatelé s OS MS Windows, je potřeba zajistit přístup i jim. Často bývá kladen požadavek, aby mohli používat pro přístup program MS Outlook. Takovou situaci řeším nainstalováním konektoru do Outlooku. Našel jsem dva komerční (za peníze) konektory, jejich připojíním/konfigurací se dále zabývám.

104.2.1. Toltec Connector

Toltec™ Connector (http://www.toltec.co.za/) je modul do programu Microsoft Outlook který umožňuje Outlooku ukládat informace (kontakty, kalendář, ...) na IMAP4 serveru. Uvádím jej zde proto abych popsal problémy a úspěchy v konfigurování IMAP4 serveru.

Autor sám na svých stránkách a v dokumentaci uvádí že na IMAP server musí umět autentikaci `STARTTLS` `AUTH=PLAIN` nebo `AUTH=CRAM-MD5`.

K dispozici je jednak starší stabilní verze 1.9x které by měl pro komunikaci stači Cyrus IMAP verze 1.5.x, a pak novější 2.0beta která vyžaduje Cyrus verze 2.2. Není ověřeno že Toltec Connector funguje i jiným IMAP serverem.

altnamespace: no

104.2.2. Konsec Konnektor

- Konsec Konnektor (http://www.konsec.com/de/)

FIXME:

Kapitola 105. Vytváření statických webů

Odkazy:

- Web template system (http://en.wikipedia.org/wiki/Web_template_system)

K prozkoumání a zpracování:

- Building a Static Site with Template Toolkit (http://www.linux-mag.com/id/2492)
- Template Toolkit ()
-
-

105.1. nanoc

Odkazy:

- nanoc (http://nanoc.stoneship.org/)

nanoc je nástroj pro generování statických webů. Je naprogramován v Ruby (../ruby/index.html) a instaluje se pomocí Rubygems

```
# gem install nanoc3
# cd /usr/local/bin
# ln -s /var/lib/gems/1.8/bin/nanoc3 .
```

* *Instalční postup je specifický pro Debian. Poslední dva příkazy.*

Donstalujeme další gemy které využijeme.

```
# gem install rack
# ln -s /var/lib/gems/1.8/bin/rackup /usr/local/bin/
```

```
# gem install test-spec
# gem install camping
# aptitude install libfcgi-dev
# gem install fcgi
# gem install memcache-client
# gem install mongrel
# gem install ruby-openid
```

Kapitola 106. Specifický software

106.1. SAS

SAS je systém pro školy který je naprogramovaný v Delphi. Běží na MS-Windows stanicích a používá databázový server Firebird 1.5. Součástí SASu je i ISAS, tedy Internetový SAS. Jedná se o PHP aplikaci sloužící k publikaci informací z databáze SASu. Jak databázový server, tak i ISAS lze provozovat na Linuxu. Následující text popisuje mé zkušenosti s instalcí ISAS a databázového serveru na Debianu Lenny.

Nejdříve instalace ISAS. Databázoví server běží na MS-Windows a my budeme jen zobrazovat.

```
# aptitude install apache2 php5-gd php5-interbase
```

* *Woody: apt-get install apache2 libapache2-mod-php4 php4-gd php4-interbase*

Soubory ISASu nahrajeme například do `/var/www/ISAS`. Odzkoušíme jestli je vše tak jak má být tím že se podíváme na stránku `server:/ISAS/_servertest.php`.

> **Poznámka:** Instalací balíčku php5-interbase v Lenny nainstalujeme klienta Firebird2.0 a podporu pro Interbase/Firebird do PHP5. V této chvíli ještě nevím, zdali tato verze klienta bude fungovat při spojení se serverem verze 1.5.

* *První experimenty napovídají že to možné je. Tedy na ISAS serveru použít v php klienta verze 2.0 pro přístup k databázi.*

Dalším krokem je přenesení databáze z MS-Windows stroje na Linuxový server. Začneme instalací databáze podle 45.3.1. Poté potřebujeme tuto databázi naplnit daty. Na MS-Windows stroje vytvoříme nešifrovanou zálohu a tuto na Linuxovém stroji naimportujeme.

```
#
```

Přeložení popdpory/klienta Firebird1.5. Budeme potřebovat balíčky ze starší verze Debain Etch. Do `/etc/apt/sources` list vložíme:

```
deb http://ftp.cz.debian.org/debian/ etch main
deb-src http://ftp.cz.debian.org/debian/ etch main
```

Stáhneme si zdroje php5.

```
/usr/src# apt-get source php5-interbase
# aptitude install php5-dev
Následující NOVÉ balíky budou instalovány:
  php5-common{a} php5-dev shtool{a}
```

Ve zdrojích vyhledáme adresář s rozšířenímy a v něm přeložíme rozšířeny interbase.

```
# cd /usr/src/php5-.../ext/interbase
# phpize
# ./configure
```

* ***phpize** mi nechtělo fungovat. Problém jsem dohledal až k problémové instalaci balíků libtool a php5-dev. Odinstalovat oba a nainstalovat v pořasí librool, php5-dev.*

```
# aptitude install firebird2-dev
```

XIII. Různá HW zařízení

DNS-323

Radek Hnilica

FIXME:pubdate

$Date: 2008/06/22 11:51:32 $

Přehled revizí

Revize $Revision: 1.37 $ $Date: 2008/06/22 11:51:32 $ Revidoval: rfh
Aktuální pracovní verze.
Revize 0 2007-07-12 Revidoval: rfh
Publikováno

Overview

This document sumarizes my own knowldge about D-Link DNS-323, the NAS device I own.

ěščřžýáíéúů

Number of pages in Postscript and PDF version: 619 .

Obsah

1. Informace a odkazy:

- DNS-323 disk activity all the time (without me doing anything) ? (http://forum.dsmg600.info/viewtopic.php?pid=18308)
-

2. Environment

When reading this document you need to know some context i which I use the nas DNS-323. And in this section I explain it.

I have a local network at my home. Its ip is 172.31.1.0/24. The DNS-323 is named as `nas1` and has ip 172.31.1.21. I note the nas1 name in my hosts file, and thus I'm able to access it as a nas1 instead of 172.31.1.21.

```
my-computer # grep nas1 /etc/hosts
172.31.1.21     nas1
```

My computer have the ip 172.31.1.8.

I have this device for archiving my data and for that reason I use RAID-1 (mirroring) configuration. I do not want to lost my data. Because of this there are two identical hard disc in the box. I buy 1TB WD disc from the RAID edition. THe are not so cheap as others, but my data are more pricey and I want have them safe.

3. How to organize your data

I have to write more about this thema. And I promise to do so if I have time. For now only few words.

The device is constructed and programmed in a way having only one main data volume, Volume_1. So if you want few volumes, I can't be done in native way. So create directory for every data export (volume) you need in the root of the Volume_1. Then export each such a directory using nfs from funplug or by original adminsitrative web interface by samba. Set the rights so user can't do a things you din't allow them.

4. Using fun_plug software

Fun plug is a way, how to extend the functionality of the box withou modifing it's firmware. So you don't broke your waranty. Also If something goes wrong, you can rename one file and efectively stop the funplug software from working. From that moment its a dead pice of data files, until you restore that one file back in it's previous state.

How it works? Very simply. After the device starts, it's operation system checks the presence of file named `fun_plug` in root of your Volume_1. If it exists, the os runs it. And thats all. This is the plug, that gives you fun to add a new functionality to your box.

4.1. Installing the FunPlug

References:

- Installing funplug-0.3 on the D-Link DNS-323 (http://techblog357.blogspot.com/2007/08/installing-funplug-03-on-d-link-dns-323.html)

First get the funplug software from www.inreto.de (http://www.inreto.de/dns323/fun-plug). I use the latest known version in time of writting this document which is 0.5. You will need the `fun_plug` and `fun_plug.tgz` files. Download these. Alse read the `README.txt`.

Now is time to gain access to dns-323. You could not use nfs or ssh as those services are not yet running. You can only use cifs (samba).

```
# mount -t cifs //nas1/Volume_1 /mnt
```

The copy both files you downoladed int root of the Volume_1

```
# cp fun_plug fun_plug.tgz /mnt/
# umount /mnt
```

And this is all. After rebooting the device the fun_plug software start and you can telnet to your box.

4.2. Securing fun plug

```
/mnt/HD_a2 # chmod -R o-rwxX ffp
/mnt/HD_a2 # chmod 0770 fun_plug
/mnt/HD_a2 # chown root:root fun_plug
```

4.3. NFS

There are two nfs servers in th funplug. One, the kernel nfs (nfsd), needs a kernel module. So I didn't test it yet. The second runs completly in user space, so no kernel module is needed and the usage is simple.

Connect to the device as a root. You can use the original telnet, or ssh if you configure it.

No you have to check if there is `exports` in `/ffp/etc` directory. I can be here from your previous experiments with nfs if there were some. So backup or simply remove that file. Now start the user space nfs server.

```
# sh /ffp/start/unfsd.sh start
```

When you start unfsd for the first time, it creates exports file. Now look in that file and modify it to meet your needs. Now your nfs server is running, but not scheduled to run after restarting on device. To do so we need to execute command

```
# chmod a+x /ffp/start/unfsd.sh
```

If you modify `/ffp/etc/exports` file, you need to restart the nfs server. Or your changes are not recognized until restarting your device. So run the command

```
# /ffp/start/unfsd.sh restart
```

Now, on your main computer, look what's really published from your nas device.

```
my-computer # showmount -e nas1
Export list for nas1:
/mnt/HD_a2 172.31.1.8
```

4.4. SSH

5. Some tasks

5.1. Reseting the device

While experimenting I brick my device. So I need to set it to factory default satate. I know that there is a reset pinhole, but it didn't work for me. So I came with another way to do it.

Reseting the device for me is two procedures.

1. Setting firmware to its factory default state.
2. Erasing disc and format new RAID.

The first task is simple. We need to get into admistrative web interface. After some searching on net I come to wollowing steps.

1. Shutdown the device if possible. This can't be done if you locked the device enough. So skip to next step.
2. Unplug the power cord. We need to be sure we do not damage electronic inside.
3. Remove front panel of the device. Slightli pushin it upvards and pulling forward from the device.
4. Remove both disc, if you have two. I always use RAID-1 (mirroring) so I have alwas two disc.
5. Plug the power cord and press the microswith on the front to startup the device.
6. After device succesfully starts, go to the web interface and login into it. THen find the TOOLS menu, select SYSTEM and press the Restore button to „Restore To Factory Default Settings." After a while the device reboots and its in the same state as you unpack it from the box.

The second procedure could ba a little harder. We need there be no information on the disc. We need the device do not recognize previous data structures. The safest way is to plug the disc into another computer, or use SATA to USB converter. The we erase the first bits of the disc with **dd** command.

```
# dd if=/dev/zero /of=/dev/sdb bs=1M count=10
```

In command shown above I assume that the attached disc is recognized as `/dev/sdb`. Check twice if its right or you can erase another disc in your system!

Now you can shutdown the DNS-323, unplug power cord and insert both disc into it. After that return back the front panel a plug and power on the device.

6. How the device works and what's in it

6.1. fun_plug

During some part of system start, I have to check which, the systems check if there are any files named `fun_plug`.

The script `/usr/sbin/chk_fun_plug` looks in the Volume_1 which is mounted on `/mnt/HD_a2`, if there is an executable file named `fun_plug`. If it find that file, the file is executed.

Příloha A. Seznam osobností z UNIX scény

Petr Vaněk

Přispívá do konference news://cz.comp.linux.debian

Prezentace: http://www.penguin.cz/~vanous

Adresa:

Adresa:

FIXME: Autor programu: (xref)

Ken Thomson

Dennis Ritchie

R.H. Canaday

Brian Kernighan

Index

D

Literatura

Knihy

[1AhoSethiUllman96] *Compilers, Principles, Techniques, and Tools*.

I. Moje Reference

as

Jméno

`as` — the assembler

DESCRIPTION

FIXME:popis.

Example Glossary

E

Extensible Markup Language

: Some reasonable definition

Viz též: Standard Generalized Markup Language.

S

SGML

: *Viz:* Standard Generalized Markup Language

Standard Generalized Markup Language

: Some reasonable definition

Viz též: Extensible Markup Language.

Tiráž

Tiráž

FIXME:napsat.