

Zálohovací server založený na GNU/Linux Debian

Backup server based on GNU/Linux Debian

Tomáš Knot

Bakalářská práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Tomáš KNOT**
Osobní číslo: **A08053**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**

Téma práce: **Zálohovací server založený na GNU/Linux Debian**

Zásady pro vypracování:

1. Provedte literární rešerši na dané téma.
2. Nainstalujte a nakonfigurujte zálohovací server na GNU/Linux Debian.
3. Naprogramujte zálohovací skript v programovacím jazyce Bash.
4. Ověřte funkčnost skriptu a jeho připojení na vzdálená zařízení.
5. Popište zabezpečení implementovaného řešení.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MASTERS, Jon; BLUM, Richard. Linux profesionálně : programování aplikací. Vyd. 1. Brno : Zoner Press, 2008. 539 s. ISBN 978-80-86815-71-8.**
2. **MATTHEW, Neil; STONES, Richard; KRÁSENSKÝ, David. Linux : začínáme programovat. Brno : Computer Press, 2008. 829 s. ISBN 978-80-251-1933-4.**
3. **NEMETH, Evi; HEIN, Trent R; SNYDER, Garth. Linux : kompletní příručka administrátora. Vyd. 1. Brno : Computer Press, 2004. 828 s. ISBN 8072269194.**
4. **PTÁČEK, Lubomír. Linux : dokumentační projekt. 4., aktualiz. vyd. Brno : Computer Press, 2007. 1334 s. ISBN 978-80-251-1525-1**
5. **AOKI, Osamu. Debian Reference [online]. 2. verze. 2010-12-10 [cit. 2011-01-07]. Dostupné z WWW: .**

Vedoucí bakalářské práce:

Ing. Jiří Korbel

Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce:

25. února 2011

Termín odevzdání bakalářské práce:

7. června 2011

Ve Zlíně dne 25. února 2011



prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Cílem práce je vytvořit zálohovací server založený na GNU/Linux Debian, který bude v určitou dobu každý den kopírovat log soubory ze zhruba 40 serverů. Server je bude ukládat a třídit podle určitých kritérií (jako je název vzdáleného zařízení a datum).

Zálohovací server bude obsahovat pevné disky zapojené do pole RAID-1 a nad tímto polem bude vytvořen oddíl LVM2. Kopírování souborů s logy bude probíhat pomocí SSH a RSA klíčů. Na serveru bude nainstalován syslog server pro logování ze vzdálených zařízení. Zálohovací skript v BASHI zajišťuje kopírování a ukládání na disk. Po provedení celého zálohování je odeslán email s průběhem a výsledkem zálohování (zda se jednotlivé logy zkopírovaly správně či nikoliv).

Přínosem tohoto řešení má zálohovací server v tom, že zálohuje logy pro případ selhání (ztráty dat) na primárním úložišti (serveru). Výhodou toho řešení je to, že veškerá data jsou na jednom místě a jsou tak v časové souvislosti lépe dohledatelná. Důležitou částí instalace je i síťový syslog server, který ukládá logy ze zařízení, jenž nemají žádný pevný disk, a proto se zprávy či hlášení ukládají pouze do paměti přístroje. V případě výpadku elektrického napájení nebo nějakého selhání jsou hlášení ztracena.

Klíčová slova: Linux, Debian, zálohování, RAID, SSH, LVM, syslog

ABSTRACT

The objective of this work is to create a backup server based on GNU/Linux Debian, which will copy log files at some time each day from around 40 servers. This server will save and sort those logs according to certain criteria such as a date and name of the remote servers.

The backup server will contain hard drives in RAID-1 and a LVM2 partition will be set up as a part of this field. Copying of log files will proceed via SSH and RSA keys. The syslog server will be installed on the backup server to log remote devices. A BASH backup script provides copying of logs and their saving to the hard disk. After finishing the backup process an email is sent with information, whether the individual logs are copied correctly or not.

Benefits from this solution is that the server logs are backed up in case when failure (data loss) of primary storage (server) occurs. The advantage of this solution is that all data are stored in one place and therefore they can be easily found in context of time. The important part of installation is the network syslog server that saves logs from the device with no hard drive installed and they have to store messages in device's memory. In the event of a power failure or some other failure the data are lost.

Keywords: Linux, Debian, backup, RAID, SSH, LVM, syslog

Rád bych poděkoval za vedení bakalářské práce panu Ing. Jiřímu Korbelovi, PhD. za pomoc a rady při psaní této práce. Mé poděkování patří také odbornému konzultantovi Ing. Albertovi Mrázkovi za praktické rady při tvorbě zálohovacího serveru.

Děkuji své rodině za podporu při studiu, kterou mi poskytovala.

„Každý, s kým se v životě potkám, mě v něčem předstihuje. Tak se od něho učím.“

Ralph Waldo Emerson

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	10
I TEORETICKÁ ČÁST	11
1 CO JE TO LINUX?	12
1.0.1 Linuxové jádro a jeho vývoj	12
1.0.2 Distribuce	12
1.0.3 Jak instalovat programy	13
1.1 HISTORIE A SOUČASNOST	13
1.1.1 Historie	13
1.1.2 Současnost	14
1.2 DISTRIBUCE DEBIAN	14
1.2.1 HW podpora	15
1.2.2 Správce balíčků Advanced Packaging Tool	15
1.2.3 Vývojové větve a životní cyklus	15
1.2.4 Debian 6.0 Squeeze	15
1.3 GNU v LINUXU	16
1.3.1 GNU GPL	16
2 ZÁLOHOVÁNÍ A JEHO DŮVODY	17
2.1 DRUHY ZÁLOHOVÁNÍ	17
2.1.1 Jednoduché zálohování	17
2.1.2 Víceúrovňové zálohování	18
2.2 PRAVIDLA ÚSPĚŠNÉHO ZÁLOHOVÁNÍ	18
2.3 KOMPRIMOVÁNÍ UKLÁDÁNÝCH DAT	18
2.4 ZÁLOHOVACÍ MÉDIA	19
3 SPRÁVA OPERAČNÍHO SYSTÉMU LINUX	20
3.1 SOUBOROVÉ SYSTÉMY	20
3.1.1 Žurnál souborového systému	21
3.1.2 Přehled souborových systémů	21
3.2 REDUNDANT ARRAY OF INDEPENDENT DISKS (RAID)	22
3.2.1 Úrovně RAID	22
3.2.2 Záložní versus poškozený disk	25
3.2.3 Rozdíl mezi hardwarovým a softwarovým RAIDem	25
3.3 LOGICAL VOLUME MANAGEMENT (LVM)	26
3.3.1 Funkce LVM	26
3.3.2 Mapovací režimy LVM	27
3.3.3 Snímky LVM (Snapshots)	27

3.3.4	Vytvoření LVM	28
3.4	IPTABLES	28
3.4.1	Vytvoření příkazu iptables	29
3.5	CRON.....	31
3.5.1	Crontab	31
3.5.2	Nastavení crontabu.....	31
3.6	SYSLOG - LOGOVÁNÍ SYSTÉMOVÝCH UDÁLOSTÍ.....	32
3.6.1	Nastavení rsyslogd	32
4	TVORBA ZÁLOHOVACÍHO SKRIPTU	33
4.1	ZASÁDY TVORBY SKRIPTŮ.....	33
4.2	PŘÍKAZOVÝ INTERPRET BOURNE AGAIN SHELL (BASH)	33
4.2.1	Příkazy pro BASH.....	34
4.2.2	Spuštění shelového skriptu.....	34
4.3	PROGRAMOVACÍ JAZYK GNU AWK	35
4.3.1	Příkazy pro awk	36
4.4	EDITOR STREAM EDITOR (SED)	36
4.4.1	Příkazy pro sed	36
4.5	SECURE SHELL (SSH)	37
4.5.1	Metody ověření identity	38
4.5.2	Man-in-the-middle (muž uprostřed)	38
4.5.3	Připojení na vzdálené zařízení pomocí ověrování klíčů.....	39
4.6	DOMAIN NAME SYSTEM (DNS)	39
4.6.1	DNS zone transfer	39
II	PROJEKTOVÁ ČÁST	41
5	DŮVODY VYTVOŘENÍ ZÁLOHOVACÍHO SERVERU.....	42
6	INSTALACE OPERAČNÍHO SYSTÉMU.....	42
6.1	VYTVOŘENÍ RAID POLE	42
6.2	VYTVOŘENÍ LVM2.....	45
7	ZABEZPEČENÍ OPERAČNÍHO SYSTÉMU.....	46
7.1	ZABEZPEČENÍ POMOCÍ IPTABLES	46
7.2	ZPROVOZNĚNÍ SSH SERVERU.....	48
7.2.1	Nastavení SSH serveru	49
7.2.2	Připojení na vzdálené zařízení pomocí hesla	51
7.2.3	Generování privátního a veřejného klíče	51
8	ÚKOLY SYSTÉMU	52
8.1	ZPROVOZNĚNÍ RSYSLOGU.....	52

8.1.1	Nastavení rsyslogu	52
8.2	ZÁLOHOVÁNÍ LOG SOUBORŮ	54
8.2.1	Zálohovací skript	54
8.2.2	Nastavení crontabu	56
	ZÁVĚR	58
	ZÁVĚR V ANGLIČTINĚ	60
	SEZNAM POUŽITÉ LITERATURY	62
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	66
	SEZNAM OBRÁZKŮ	66
	SEZNAM TABULEK	68
	SEZNAM PŘÍLOH	69

ÚVOD

Jedním z nejdůležitějších úkolů správce jakéhokoliv serveru je pravidelné zálohování lokálních systémů nebo vzdálených zařízení. Zálohovány jsou data, která mají charakter vysoké důležitosti a jsou nezbytná v dané oblasti působnosti serveru nebo vzdáleného zařízení. V případě náhlého poškození serveru či jiné neočekávané události, je možné kritická data obnovit do původního stavu.

V oblasti serverů má své nezastupitelné místo operační systém Linux. Systém je velmi oblíben díky své otevřenosti, stabilitě a nákladům. Velkou výhodou je i modularita systému, kdy je možné jej upravit do požadované podoby. Z těchto důvodů byl pro tvorbu zálohovacího serveru vybrán GNU/Linux Debian.

Cílem této práce bude vytvoření zálohovacího serveru. Server se bude starat o zálohování log souborů ze zhruba 40 vzdálených zařízení, která jsou rozmístěna v různých lokalitách. Zálohování bude probíhat denně ve stanovený čas pomocí skriptu naprogramovaného v příkazovém interpretu BASH. Skript se bude starat o ukládání logů na server podle zadaných kritérií (název vzdáleného zařízení, rok, měsíc). Velikost log souboru pro jedno vzdálené zařízení se bude pohybovat v závislosti na provozu sítě okolo 50MB.

Součástí zálohovacího serveru bude část starající se o uchovávání hlášení ze vzdálených zařízení. Velkou výhodou bude mít v případě zařízení, které mají uloženy hlášky a log soubory v paměti a neobsahují pevný disk. Tyto údaje mohou být bezpečně uchovány v případě ztráty napájení nebo jiné neočekávané události. Tuto činnost bude zajišťovat síťový syslog server. Data budou uchována na discích, které budou umístěny v poli RAID-1. Nad částí tohoto pole bude vytvořen oddíl LVM2.

Výsledkem této centralizace log souborů bude snadné dohledávání různých časových souvislostí se správou jednotlivých vzdálených zařízení. Tyto log soubory mají charakter systémových údajů a je nutné provádět pravidelnou zálohu pro případ dalšího využití při správě datové sítě.

Tato práce bude řešena pro konkrétního poskytovatele internetového připojení ve Zlínském kraji. Veškeré názvy počítačů a jejich IP adres jsou smyšlené z důvodu bezpečnosti zálohovacího serveru, na kterém se nachází citlivé údaje pro správu datové sítě této společnosti.

I. TEORETICKÁ ČÁST

1 CO JE TO LINUX?

V dnešní době se lze setkat s Linuxem ve všech oblastech, i když o tom ani mnozí lidé netuší. Nachází se od mobilního telefonu, přes osobní počítače, routery, servery až po výkonné výpočetní systémy. Linux je operační systém (OS) unixového typu. Je to volně šiřitelný software tzv. open source software. Uživatel může šířit a upravovat zdrojové kódy. Díky těmto vlastnostem se stal Linux populární v oblasti serverů, kde má své pevné postavení.

1.0.1 Linuxové jádro a jeho vývoj

Za vývojem jádra OS stojí velké softwarové firmy jako IBM, Red Hat, Google, Novell apod., ale také řada nezávislých vývojářů. Linuxové jádro podporuje různé druhy hardwarových (HW) architektur. Namátkou jsou to Intel i386, amd64, PowerPC, Intel IA-64, ARM a další. Jádro je dodáváno ve formě zdrojových kódů, které je nutné kompilovat pro daný HW. Kompilace má výhodu v tom, že sestavení odpovídá jak HW požadavkům, tak i požadavkům uživatele. Jelikož proces kompilování je zdlouhavý, tak uživatelé nejčastěji využívají formu linuxových distribucí.

Linuxové jádro není součástí GNU projektu. Z tohoto projektu je využita jen licence. Systém obsahuje jádro a další nástroje (nejsou specificky linuxové), které jsou z projektu GNU. Proto by se systém měl označovat GNU/Linux. Nejnovější verzi jádra lze nalézt na <http://www.kernel.org>. [1]

1.0.2 Distribuce

Linux je ve větší části šířen jako distribuce. Distribuce se skládá ze samotného linuxového jádra, grafického rozhraní a dalšího softwaru, který je uživatelům předkládán v předem připravené podobě. Existuje velké množství distribucí, které mají různé zaměření od serverových, desktopových až po embeddend zařízení. [2] Mohou být rozděleny na dvě části - komunitní a komerční distribuce. Komunitní distribuce jsou vyvíjeny komunitou uživatelů či firmou, která podporuje vývoj. Mezi nejznámější komunitní distribuce patří GNU/Linux Debian, Fedora, Ubuntu, OpenSUSE, jež jsou dodávány bez placené uživatelské podpory. Uživatelskou podporu si zajišťuje svépomocí uživatel např. pomocí různého komunitního fóra. Pokud by byl Linux využit u komerčních řešení, je běžné využití komerční distribuce, která nabízí placenou uživatelskou podporu. Jsou to např. Red Hat Enterprise Linux, SUSE Linux Enterprise. Zde firmy zajišťují plnou uživatelskou podporu. Výrobci se snaží své produkty certifikovat pro různý serverový HW, kde garantují bezproblémový chod svého softwaru.

1.0.3 Jak instalovat programy

Instalace programů je zajištěna pomocí repozitářů. V těchto repozitářích jsou na serveru uloženy zkompileované programy, které uživatel může instalovat. Tuto instalaci provádí tzv. správce balíčků, zajišťující vše potřebné pro správnou instalaci. Pokud program potřebuje určitou knihovnu nebo jiný program, který není nainstalován v počítači, tak si jej balíčkovací systém dodatečně nainstaluje. Programy lze instalovat i bez repozitářů jako samostatné balíčky. Mezi nejznámější balíčky patří *.deb* a *.rpm*. Každá distribuce využívá některý z typů balíčků. Některé programy jsou šířeny ve formě zdrojových kódů. Pro jejich spuštění je nutné nejprve provést kompilaci, která zajistí překlad do binární podoby.

1.1 Historie a současnost

1.1.1 Historie

Poprvé se Linux objevil v roce 1991, kdy jeho autor Linus Torvalds, toho času student helsinské univerzity, začal vyvíjet jádro nejprve jako své hobby. Svou inspiraci našel v systému Minix (systém UNIXového typu, autor Andy Tanenbaum). Cílem bylo vytvořit systém, jehož parametry by splňovaly standard POSIX. Na konci srpna roku 1991 byla vydána první verze 0.01, která ještě nebyla veřejná. První veřejnou verzí byla 0.02 (oznámena přes diskuzní skupinu *comp.os.minix*). V dalším průběhu let byly verze uvolňovány až po dnešní verzi 2.6. Důležitou součástí Linuxu jsou i GNU nástroje (**make**, **wget**, **less** apod.) vytvořené nadací Free Software Foundation. [3]



Obr. 3. Maskot Linux tučnák Tux [4]

1.1.2 Současnost

V současnosti se vývoj nachází ve verzi 2.6.39 (květen 2011). Správcem celé větve je sám Linus Torvalds, který se zaměřuje na vývoj nových verzí. O jednotlivé starší větve se starají pověření správci. Některé verze jádra jsou označeny jako longterm, které mají prodlouženou dobu podpory. Doba podpory závisí na správci dané větve. Většinou se jedná o podporu v rozmezí zhruba 2 - 3 let. Po uplynutí se o podporu stará distribuce, která jádro využívá. Pro vývoj se používá jazyka C. U případů, kdy je potřeba komunikace s HW na nejnižší úrovni, je využíván assembler (jazyk symbolických adres). [5]

1.2 Distribuce Debian

Zakladatelem distribuce Debian je Ian Murdock. Založena byla 16. srpna 1993. Název pochází ze dvou jmen **Debra** (manželka Iana Murdocka) a **Ian** (jméno zakladatele). Současná verze je Debian 6.0 Squeeze (jména verzí pochází z filmu Toy Story - Příběh hraček). Nová verze vychází zhruba každé dva roky. Debian se zavázal Společenskou smlouvou, ve které se zaručují dané vlastnosti a jejich dodržení. [6, 7]

- Distribuce bude vždy otevřená a zdarma
- Bude vždy vracet kódy komunitě
- Neskrývá problémy při vývoji a v organizaci
- Zůstane zaměřena na uživatele a software
- Software řeší reálné problémy



Obr. 4. Logo distribuce [8]

Jedná se o svobodnou distribuci, která je vyvíjena za pomoci dobrovolníků z celého světa. Za vývojem stojí zhruba 1030 vývojářů a další tisíce dobrovolných přispěvatelů. Vývoj Debianu nezajišťuje žádná firma a tím pádem je jeho vývoj decentralizovaný. Opačným příkladem vývoje je distribuce Ubuntu, kterou podporuje firma Canonical. Z Debianu vychází mnoho dalších distribucí např. Ubuntu, Mint, Knoppix a další. Distribuci Debian lze stáhnout z webových stránek <http://www.debian.org>. [7]

1.2.1 HW podpora

Distribuce podporuje HW architektury Intel i386, amd64, PowerPC, ARM a další. Celkem je podporováno deset linuxových a dvě FreeBSD architektury. Základem systému je jádro GNU/Linux, ale je možné použít i nelineková jádra typu Hurd, NetBSD, kFreeBSD. [7]

1.2.2 Správce balíčků Advanced Packaging Tool

Pro instalaci a správu balíčků se používá správce Advanced Packaging Tool (APT). Balíčky mají příponu *.deb* a je to archiv typu *ar*. Instalace balíčku se provádí za pomoci přímo správce balíčků APT nebo jeho nadstaveb, které volají instalátor (program) *dpkg*. V Debianu se pro instalaci balíčků z repozitářů doporučuje využívat konzolové nadstavby Aptitude. [7]

1.2.3 Vývojové větve a životní cyklus

Debian je znám svou konzervativností a stabilitou. Proto vývoj probíhá ve třech různých větvích - *stable*, *testing* a *unstable* + *experimental*. Starší vydání je přesunuto z větve *stable* do *oldstable*. *Stable* je základní větev. Zde je veškerý software stabilní a odladěný. Na tyto vlastnosti je kladen vysoký důraz, protože verze *stable* je nasazována na produkčních zařízeních, kde je požadována stabilita. Nenachází se zde aktuální verze softwaru, které jsou v jiných větvích. *Testing* zajišťuje odladění a testování softwaru pro *stable*. Jsou zde odladěny veškeré chyby. Zmrazením *testing* větve je po určité době od zmrazení vydána *stable* verze Debianu. Zde se nachází již novější software. Poslední je vývojová větev *unstable*. Ve větvi je nejaktuálnější software, proto může být používání někdy mírně problematické. Existuje také větev *experimental*, kde se nachází neotestované balíčky, které budou později zařazeny do *unstable*. [7]

V průběhu životního cyklu prochází balíček všemi větvemi od zařazení do *experimentalu* přes *unstable*, *testing* až po finální větev *stable*. Každá větev má své jméno mimo *experimental*. *Unstable* je nazýván *Sid*. Toto jméno je trvalé pro všechny verze Debianu. *Stable* má jméno pro verzi Debian 6.0 Squeeze a *testing* se jmenuje Wheezy. Budoucí *stable* bude Wheezy a *testing* se pojmenuje novým jménem. [7]

1.2.4 Debian 6.0 Squeeze

Debian 6.0 Squeeze byl vydán 6. února 2011 po dvou letech od předchozího vydání. Základem je jádro verze 2.6.32-5. Dále obsahuje grafické prostředí Gnome 2.30.2, KDE 4.4.5, XFCE 4.6.2 a LXDE 0.5.0. Novinkou je použití zavaděče Grub2 a souborového systému ext4. Z Debianu 5.0 Lenny se stává *oldstable* a jeho podpora bude ještě trvat

tři roky. Nově vzniknul testing se jménem Wheezy. [9]

1.3 GNU v Linuxu

Linux je postaven na kultuře volně šiřitelného softwaru. Má otevřený zdrojový kód, je bezplatný a jeho využívání není omezeno. Jediným omezením je licence GNU General public licence (GPL). Licence je pod záštitou nadace Free Software Foundation (FSF). [10]

Zakladatel FSF Richard Stallman založil také GNU Project. Tento projekt si klade za cíl vytvořit operační systém unixového typu, jenž nebude svázán licenční politikou Unixu. Linuxové jádro není součástí GNU, ale využívá stejnou licenci. GNU Project je soubor programů, které jsou podobné aplikacím na Unixu. Mezi nejdůležitější příklady softwaru distribuovaného projektem GNU pod podmínkami GPL patří: [1, 10]

- GCC: The GNU Compiler Collection, který obsahuje kompilátor GNU C
- G++: kompilátor jazyka C++, který je součástí GCC
- GDB: debugger na úrovni zdrojového kódu
- GNU make: verze unixového nástroje make
- Bison: generátor lexikální analýzy kompatibilní s unixovým programem yacc
- bash: příkazový interpret neboli shell
- GNU Emacs: textový editor a prostředí
- The GIMP: grafický program pro rastrovou grafiku

1.3.1 GNU GPL

GNU GPL je licence, která se stará o nakládání se softwarem a jeho zdrojovým kódem. Uživatelům dovoluje úpravu programu a další šíření upravené verze. Tato úprava musí splňovat podmínky GNU GPL licence. Program může být také šířen za poplatek, ale musí být opět přístupný podle licence. Zpoplatnění může být způsobeno náklady na výrobu a distribuci datového nosiče ke koncovým zájemcům. [3]

Distributor musí dát vždy vědět, že program je chráněn podmínkami GPL. Dále nesmí omezovat práva kupujícího a musí poskytnout kompletní zdrojové kódy distribuovaného programu. Kupující může se softwarem nakládat podle GPL, tedy šířit, kopírovat, distribuovat bezplatně nebo za poplatek. [3]

Oficiální text licence je v angličtině a lze ho nalézt na webových stránkách GNU Project <http://www.gnu.org/licenses/gpl.html>. V současné době se licence nachází ve verzi GPLv3. Linux stále ještě využívá licenci verze GPLv2 viz <http://www.kernel.org/pub/linux/kernel/COPYING>.

2 ZÁLOHOVÁNÍ A JEHO DŮVODY

Všechny kritické systémy nebo důležitá nenahraditelná data musí být pro případ havárie nebo poškození HW či jiné chyby zálohovány. Zálohování je proto velmi důležité, neboť dnes mají data větší cenu než HW samotných počítačů, a proto je i ochranou investic do dat a informací. Příkladem jsou firmy, pro které jsou data životně důležitá. Patří sem výrobní podniky, poskytovatelé telekomunikačních služeb, nemocnice a další.¹⁾ Důležitou roli hraje i kvalita zálohy (doba obnovy, pravidelnost zálohovacího cyklu), ale i kontrola zda je záloha funkční. [1, 12]

Neméně důležitou činností je fyzické a softwarové zabezpečení záloh. Pokud se jedná o citlivá data, která musí být uchovávána podle určitých pravidel (způsob uchování je dáno zákonem apod.) je na místě jejich zabezpečení.

2.1 Druhy zálohování

Zálohy se dělí na dvě kategorie - jednoduché a víceúrovňové. Každá z těchto úrovní má své výhody a nevýhody a lze je mezi sebou kombinovat. [1]

2.1.1 Jednoduché zálohování

Jednoduché zálohování má dvě úrovně - úplné a inkrementální. V první fázi dochází k úplnému zálohování, kdy jsou veškerá data zálohována. Poté následuje inkrementální záloha, která již provádí pouze změny od poslední úplné zálohy (budou zkopírovány jen ty soubory, u kterých došlo ke změně). Tento způsob je vhodný pro osobní potřebu nebo systémy s malým počtem uživatelů. Historie zálohy sahá pouze k poslední úplné záloze. [1]

Příkladem může být kombinování těchto dvou způsobů, kdy např. ve firmě je potřeba provést zálohu. V jeden určený den bude provedena úplná záloha a každý následující den se vykoná pouze inkrementální záloha do další úplné zálohy. Nejprve je nutné provést úplnou zálohu před smazáním předešlé. Mohlo by se stát, že by byl celý proces k ničemu. [1]

¹⁾Podniky v Česku a Švédsku nejvíce pravidelně externě zálohují data ze všech zemí EU27 (březen 2011, Computerworld) <http://computerworld.cz/aktuality/csu-podniky-v-cesku-ve-vyuziti-pocitacovych-siti-za-staty-eu-stale-zaostavaji-42949>

2.1.2 Víceúrovňové zálohování

Víceúrovňové zálohování staví na více úrovních oproti jednoduchému zálohování, které lze zobecnit do libovolného počtu úrovní. Je to kombinace jednoduchého zálohování. Úplná záloha je na úrovni 0. Další úroveň je inkrementální a odpovídají jí úrovně 1, 2, 3 a dále. V každé úrovni inkrementálního zálohování se zálohují změny, které jsou rozdílné oproti poslední záloze na stejné nebo nižší úrovni. [1]

Výhodou tohoto řešení je levné prodloužení historie zálohovaných dat a snižuje čas potřebný k obnově dat. [1]

2.2 Pravidla úspěšného zálohování

Dobrá záloha musí být kvalitní a proto musí splňovat určité náležitosti. Následná pravidla jsou obecná, která udělají proces zálohování hladším. Dodržením těchto pravidel je možné se vyvarovat chyb. [12]

- Mít určeno co se bude záloovat
- Interval doby zálohování
- Volba zálohovacího média
- Druh zálohování
- Volba zálohovacího programu (skriptu nebo způsobu)
- Zálohovat vždy na jiné zařízení než z kterého je zálohováno
- Provádět kontrolu záloh

2.3 Komprimování ukládaných dat

Při zálohování jsou přenášeny velké objemy dat. Místo na které jsou informace ukládány není nekonečně velké. Je vhodné provádět komprimaci záloh, kdy je jejich velikost zmenšena v závislosti na komprimačním algoritmu. V Linuxu se lze setkat standardně s jednou archivační a dvěma komprimačními metodami. Jsou to *tar*²⁾, *gzip*³⁾ a *bzip*⁴⁾. [13, 14, 15]

²⁾Má příponu *.tar*. Nejedná se přímo o komprimační, ale o archivační program. Vytváří balík souborů, který je následně komprimován programy *gzip* a *bzip*.

³⁾Má příponu *.gz*. Využívá komprimační algoritmus Lempel-Ziv (LZ77). Nejčasteji se vyskytuje s archivačním programem *tar* (přípona *.tar.gz*).

⁴⁾Má příponu *.bz2*, *.bz*. Je to komprimační program, který je založen na Burrows-Wheelerově transformaci pro blokové třídění textového kompresního algoritmu a následné aplikaci Huffmanova kódování. Používá se i opět v kombinaci s *tar* a má příponu *.tbz2*, *.tbz*, *.tar.bz2*, *.tar.bz*. Komprimační poměr je lepší než u algoritmů LZ77/LZ78.

Bohužel se u komprimovaných záloh může vyskytnout komplikace. Z principu funkce komprimování může nastat problém. Pokud dojde k zápisu jednoho bitu špatně, je pak nepoužitelný zbytek komprimovaných dat. Z toho důvodu mají zálohovací programy zabudovány vestavěné opravné algoritmy, ale žádný z algoritmů si neumí poradit s větším počtem chyb. [1]

Alternativou je komprimace jednotlivých záloh (souborů) a v případě poškození lze využít jinou zálohu. Tímto řešením se snižuje pravděpodobnost úplné ztráty zálohy. [1] Komprimování souborů zabírá určitý procesorový čas a v případě zálohování velkých bloků dat může trvat i hodiny. Proto je nutné si plánovat dobu, kdy bude probíhat záloha.

2.4 Zálohovací média

Dnes na trhu existuje řada zálohovacích zařízení a médií, na které lze ukládat zálohy. Při výběru vhodného média hraje roli pořizovací cena, kapacita a rychlost. S rychlým médiem se příjemně pracuje a umožňuje vyšší flexibilitu v plánování záloh. [1]

- **Pevný disk** - Dnes nejčastější způsob ukládání zálohovaných dat. Je dobrým a levným řešením pro ukládání dat po síti. Disk lze sdílet pomocí Network File System (NFS) nebo Common Internet File System (CIFS). Sdílení se využívá právě u síťových zařízení Network Attached Storage (NAS). [12]
- **Optický disk (CD-R, CD-RW, DVD-R, DVD-RW, Blue-ray)** - Na CD disk lze uložit 700MB, DVD 4.7GB a Blue-ray 25GB - 80GB (záleží na počtu vrstev). Na média se zapisuje laserem. CD-R a DVD-R jsou spíše vhodné pro archivaci než pro zálohování, protože nejsou přepisovatelné. CD-RW a DVD-RW jsou vhodnější pro zálohování. Média, na která se zapisuje pouze jednou, mají menší životnost než média lisovaná. Životnost médií je delší než u magnetických médií. [12]
- **NAS** - Pole pevných disků připojených do lokální sítě. Připojení zajišťuje server pomocí síťových protokolů.
- **Magnetická páska** - Spolu s pevným diskem nejpoužívanější způsob zálohování. Páska je vhodná i pro archivaci. Velikost pásky se liší na jejím typu. (např. osmimilimetrové kazety, kazety DAT, pásky Travan, DLT, AIT apod.) [12]
- **Vzdálené zálohování přes internet** - Rozšíření vysokorychlostního internetu umožnilo vzdálené zálohování, které nabízejí různé firmy. Zde je většina záloh stále uložena lokálně. [12]

- **Ostatní média** - Využívání pamětí typů flash (USB flash paměti apod.), případně ukládání na Solid-state Driver (SSD). [12]

3 SPRÁVA OPERAČNÍHO SYSTÉMU LINUX

3.1 Souborové systémy

Souborový systém je tvořen metodami a strukturami dat. V nich operační systém udržuje záznamy o souborech na discích a diskových oblastech. Soubory jsou tedy ukládány do souborového systému, což je jistá sekce pevného disku (nebo jiného paměťového média - CD-ROM apod.) naformátována tak, aby mohla uchovávat soubory. Rozdíl mezi diskovou oblastí a diskem je značný. Jen některé programy (mezi nimi jsou programy pro tvorbu souborového systému) pracují přímo se sektory disku nebo diskovými oblastmi. Většina aplikací pracuje se souborovým systémem. Proto je nelze použít v diskové oblasti, kde není vytvořen souborový systém nebo je vytvořen jiný typ souborového systému. [1, 3]

Každý souborový systém je spjat s jinou částí adresářového stromu. Například ve většině systémů je jeden souborový systém v adresáři `/usr`, jiný v adresáři `/tmp` atd. Hlavní (kořenový) souborový systém je primárním souborovým systémem a odpovídá nejvyššímu adresáři `/`. Pro OS Linux je každý souborový systém v samostatné diskové oblasti pevného disku. V případě potřeby může být například jeden souborový systém `/` a jiný pro `/usr`, pak bude potřeba mít dvě samostatné diskové oblasti. [3]

Před instalací Linuxu je nutné připravit souborový systém pro použití programového vybavení. Proto je provedena inicializace - zápis nutných datových struktur. Tento proces je označen jako vytvoření souborového systému. [1, 3]

Většina unixových souborových systémů má podobnou obecnou strukturu. V dalších podrobnostech se celkem dost liší. Mezi důležité pojmy patří superblok, inode, datový tok, adresářový blok a nepřímý blok. Superblok obsahuje informace o souborovém systému jako celku, například jeho velikost (právě u této položky závisí přesné hodnoty na konkrétním souborovém systému). Inode obsahuje všechny informace o souboru kromě jeho jména. Jméno souboru je uloženo v adresáři společně s odpovídajícím číslem inode. Adresářová položka obsahuje jména souborů a čísla inodů, které tyto soubory reprezentují. Inode dále obsahuje čísla datových bloků, v nichž jsou uložena data souboru, který inode zastupuje. V inode je místo jenom pro několik čísel datových bloků. Když je jich potřeba více, je dynamicky alokováno více místa pro další ukazatele na datové bloky. Tyto dynamicky alokované bloky jsou uloženy v nepřímém bloku. Jak jejich název naznačuje v případě, že je potřeba najít datový blok, musí se nejdřív najít jeho číslo v nepřímém bloku. [1]

3.1.1 Žurnál souborového systému

Souborový systém, který používá žurnál (záznam o prováděných operacích) se nazývá žurnálovaný. Do žurnálu se zapisuje vše, co se přihodilo v systému. Požadavek pro zápis se objeví při operaci se souborovým systémem a úpravy se nejprve zapíší do žurnálu. Po dokončení aktualizace žurnálu se zapíše záznam „uložit“, který označuje konec položky. Pak je modifikován normální souborový systém. V případě havárie se může žurnál použít k rekonstrukci dokonale konzistentního souborového systému. Ztráta dat je pak méně pravděpodobná. [1, 12]

Žurnál je již standardní výbavou linuxových souborových systémů. Není vhodné podléhat pocitu bezpečí, že žurnál je všespasitelný. K chybě může přijít kdykoliv proto je vhodné data zálohovat. [1]

3.1.2 Přehled souborových systémů

Linux podporuje různé druhy souborových systémů. Zde je zkrácený přehled nejpoužívanějších souborových systémů.

- **Second Extended Filesystem (ext2)** - Verze bez žurnálu, která je kompatibilní se staršími verzemi. Při používání není potřeba měnit stávající souborové systémy. [1]
- **Third Extended Filesystem (ext3)** - Má všechny stejné vlastnosti jako ext2. Přidána byla podpora žurnálu. Zvýšil se výkon a doba zotavení v případě havárie systému. Stal se oblíbenějším než ext2. Byl uveden v jádru verze 2.2 a 2.4 (rok 1999). Je výchozím souborovým systémem pro Debian 5.0 Lenny. [1, 16, 17]
- **Fourth Extended Filesystem (ext4)** - Ext4 je na pokročilejší úrovni než ext3. Zahrnuje škálovatelnost a spolehlivost pro velké souborové systémy (64 bit) v souladu s rostoucí diskovou kapacitou zařízení. Je kompatibilní s přechodí verzí ext3 a opět podporuje žurnál. V současné době je výchozím souborovým systémem pro Debian 6.0 Squeeze. [18, 19]
- **ReiserFS** - Velmi robustní souborový systém, který používá žurnál. [1]
- **V File Allocation Table (vfat)** - Rozšíření souborového systému FAT známého pod jménem FAT32. Podporuje větší velikosti disků než FAT. [1]
- **Server Message Block File System (smbfs)** - Známý jako cifs. Síťový souborový systém umožňující sdílení souborových systémů s počítači MS Windows. Je také kompatibilní s protokoly pro sdílení souborů ve Windows. [1]

- **Network File System (NFS)** - Síťový souborový systém, který umožňuje sdílení souborových systémů mezi více počítači. Důvodem je zajištění jednoduššího přístupu k souborům ze všech počítačů. [1]

3.2 Redundant Array of Independent Disks (RAID)

RAID⁵⁾ je způsob jak spojit několik nezávislých disků do jednoho pole (v systému se tváří jako jedna logická jednotka nebo disk). Pole RAIDu může být konstruováno různými způsoby a to vede k různým vlastnostem v závislosti na konečné konfiguraci. [1, 20]

Využití tohoto spojení je vhodné pro případy, kdy je potřeba zajistit odolnost dat proti výpadku jednoho disku, uložením dat redundantně (v závislosti na typu RAIDu) na více disků najednou. Redundance dat snižuje velikost úložného prostoru. [1, 21]

Důležité je vědět, že RAID není žádnou obdobou zálohování a není proto možné se na něj spoléhat jako na primární zdroj zálohování. Při selhání systému může dojít k poškození řadiče disku či samotného disku apod. Proto je nutné provádět zálohy!

3.2.1 Úrovně RAID

Definováno bylo 5 typů diskových polí (RAID-1 až RAID-5). Všechny typy jsou odolné proti výpadku, ale každý z nich nabízí jiné vlastnosti, výhody a výkonnost. K těmto pěti úrovním ještě přibyl doplněk v podobě neredundantního pole označovaného jako RAID-0. Ne všechny typy jsou využívány. Některé úrovně byly důležité pro výzkumný účel a jiné nemohly být ekonomicky zrealizovány. [1, 20]

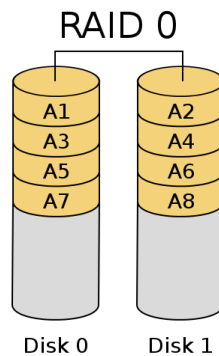
Některé původní úrovně (RAID-2 a RAID-3) jsou používány pouze ve velmi specializovaných systémech. Přibyla „lineární“ úroveň a dále je také často využíváno kombinace úrovně RAID-0 s RAID-1. Linux nepodporuje všechny úrovně, a proto pro ně neexistuje ovladač. Podporované úrovně jsou popsány v seznamu. [1]

Nejpoužívanější úrovně jsou RAID-0, RAID-1 a RAID-5. Využívá se také kombinace jednotlivých úrovní mezi sebou. [20]

- **Lineární mód** - Dva a více disků je spojeno za sebe. Zápis na zařízení probíhá lineárně. Neprve je zapsáno na disk 0 a po jeho zaplnění se zapisuje na disk 1 a dále. Velikost disku nemusí být stejná a již z podstaty zapojení není žádná redundance. Operace pro čtení a zápis se nijak nezvyšují. [1]
- **RAID-0** - Známý také jako „stripe“ mód (prokládání). Data jsou rozdělena střídavě mezi dva a více disků z pole. Opět jako u lineárního módu zde není re-

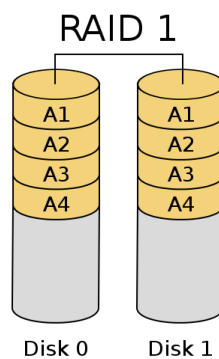
⁵⁾V počátcích vývoje se používal název *Redundant Array of Inexpensive Disks*. První zmínka o využití RAIDu pochází z Kalifornské univerzity z Barkley (1987). Dnes je znám pod zažitým názvem *Redundant Array of Independent Disks*. [20]

dundance a stejná velikost disků není nutná. V případě výpadku jsou ztracena všechna data. Operace pro zápis a čtení se zrychlí, protože probíhají paralelně z několika disků. Má nejlepší výkon ze všech RAIDů. Velikost pole je rovna počtu disků. [1]



Obr. 5. Diskové pole RAID-0 [22]

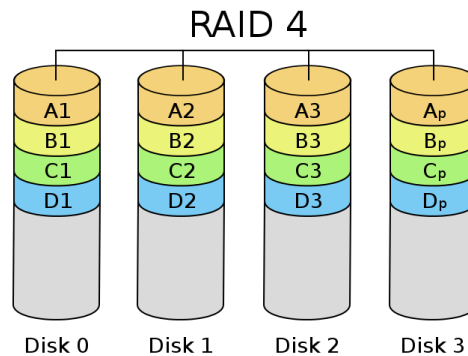
- **RAID-1** - Splňuje redundanci a dá se použít na dvou a více discích s žádným nebo více volnými (záložními) disky. RAID-1 udržuje přesnou kopii informací (provádí zrcadlení) z jednoho disku i na ostatní disky v poli. Rychlost čtení je dobrá, protože se čte z více disků současně. Zápis je pomalejší, jelikož dochází k zápisu dat na každý disk v poli. Velikost disků by měla být stejná. Pokud není dodržena velikost, je maximální velikost dána nejmenším diskem v poli. Pole udrží data při selhání N-1 disků (kde N je počet disků v poli). [1, 20, 21]



Obr. 6. Diskové pole RAID-1 [22]

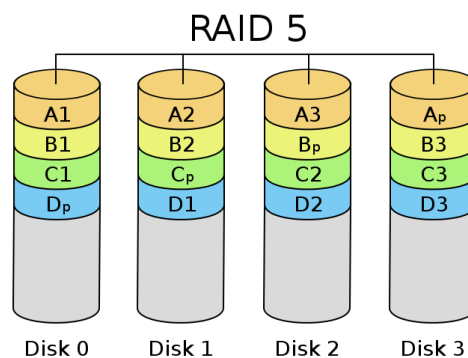
- **RAID-4** - Není příliš používanou variantou. Využití je pro tři a více disků. Místo zrcadlení disků se na jednom z nich udržují pouze paritní bity a na zbylé disky se zapisuje jako u RAID-0. Velikost pole je dána $(N-1)*S$ (kde N je počet disků, S je velikost nejmenšího disku z pole). Opět by disky měly mít stejnou velikost jako RAID-1, jinak je velikost dána nejmenším diskem z pole. Paritní disk a jeho paritní bity slouží k obnově všech dat. V případě pádů dvou disků jsou všechna

data ztracena. Důvod, proč se často nepoužívá, je ten, že paritní informace je uložena na jediném disku. Každá změna na ostatních discích musí být uložena na paritní disk a výkonnost paritního disku není výraznějsí než u ostatních. [1, 20, 21]



Obr. 7. Diskové pole RAID-4 [22]

- **RAID-5** - Výhodné použití je v případě, kdy bude k dispozici velké množství fyzických disků a je potřeba mít redundanci. RAID-5 se využívá u třech a více disků s žádným nebo více záložními disky. Velikost pole bude $(N-1)*S$ (kde N je počet disků, S je velikost nejmenšího disku z pole). Data a paritní informace jsou rovnoměrně ukládány mezi všechny disky. Operace čtení a zápis je rychlejší, ale je složité předpovědět jak moc. Čtení je stejně rychlé jako u RAID-0, zápis může být náročnější (pro správnou kalkulaci paritní informace je nejdříve nutné přechíst data z ostatních disků před vlastním zápisem) nebo stejný jako u RAID-1. Pokud selže jeden z disků, tak pomocí paritních informací zůstanou data uložena v pořádku. Všechna data jsou ztracena při selhání dvou a více disků. [1, 20, 21]



Obr. 8. Diskové pole RAID-5 [22]

3.2.2 Záložní versus poškozený disk

Čas od času se může stát, že HW vypoví službu. Pro tyto nepříjemné chvíle je vhodné využít v poli záložní disk. Jedná se o volný disk v diskovém poli, který není aktivní. V případě selhání aktivního disku je tento disk označen jako nefunkční a okamžitě začíná rekonstrukce dat na první volný záložní disk. RAID ovladač začne číst ze všech zbývajících disků, aby mohl vytvořit redundantní informace. [1]

Zvládne-li ovladač RAID výpadek bez komplikací je disk označen jako vadný. Na první volný disk začíná rekonstrukce. Poškozený disk je stále součástí pole a lze jej i vidět. Ovladač RAID jej vidí a má k němu vztah jako k neaktivní části souborového systému. [1]

3.2.3 Rozdíl mezi hardwarovým a softwarovým RAIDem

Pro vytvoření RAID pole lze využít dvou způsobů zapojení disků do společného pole. Jedná se o hardwarový a softwarový RAID.

- **HW RAID** - Pro vytvoření pole založeného na HW RAIDu se používá speciálního řadiče disku zapojeného do základní desky počítače. Řadič vykonává všechny funkce spojené s RAIDem a přímo řídí všechny jednotlivé disky zapojené do pole. Se správným ovladačem je pole spravováno HW RAIDem, který se objeví v systému, jako kdyby byly připojeny klasické disky. Většina řadičů spolupracuje se zařízeními typu SCSI, ale také i se zařízeními ATA. Administraci lze ovládat pomocí příslušného programu v operačním systému počítače, kde software vytváří rozhraní řadiče. Další možností je použití seriového portu karty přes emulátor terminálu nebo využití BIOS rozhraní během zapnutí počítače. [20]
- **SW RAID** - Je implementace RAIDu na úrovni jádra nebo softwarového ovladače pro daný operační systém. Jako takový poskytuje větší flexibilitu v podpoře HW tak dlouho, dokud je OS podporován. Umožňuje i nastavení ovládání. Výhodou je to, že odpadá nákup drahého specializovaného řadiče disku a tím se sníží finanční náklady na nasazení RAIDu. Přebytek procesorového času je spotřebováván SW RAIDem pro výpočet parity a převyšuje výkon procesoru přítomného na specializovaném řadiči. Některé SW RAIDy mají vyšší výkon než některé implementace HW RAIDů. SW RAID má omezení a ne vždy je vhodný pro nasazení na místo HW RAIDu. [20]

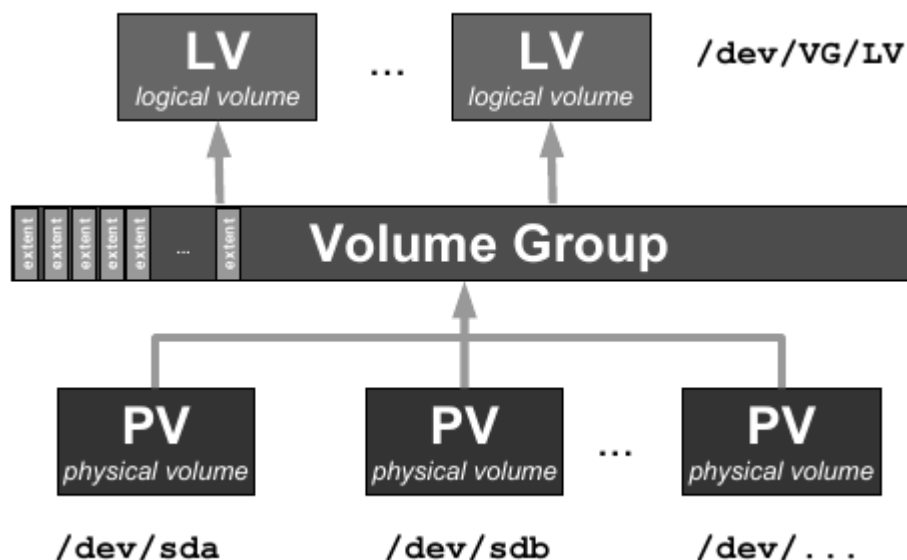
3.3 Logical Volume Management (LVM)

LVM⁶⁾ je systém pro správu logických oddílů, který využívá v Linuxu implementaci pomocí *device mapperu* (mechanismus pro mapování jednoho blokového zařízení na jiné). Umožňuje seskupovat jednotlivé disky do „svazkových skupin“. Řízení logických svazků umožňuje: [12, 23]

- Účinněji používat a alokovat diskový prostor.
- Přesouvat logické svazky mezi fyzickými zařízeními.
- Za běhu zvětšovat a zmenšovat velikost svazků.
- Vytvořit „snímek“ (snapshot) celých souborových systémů.
- Vyměňovat disky on-line bez přerušení poskytovaných služeb.

3.3.1 Funkce LVM

Ve spodní části se nachází *physical volume* (PV, fyzický svazek). Jsou to většinou pevné disky nebo jiná bloková zařízení. Nad všemi blokovými zařízeními je vytvořen *volume group* (VG, skupina svazků), který je spojí do jednoho velkého spojitého prostoru. Z toho prostoru je možné alokovat prostor pro jednotlivé *logical volume* (LV, logické svazky). S LV lze pracovat běžně jako s jakýmkoliv blokovým zařízením. [23]



Obr. 9. LVM2 - hierarchie jednotlivých vrstev [24]

⁶⁾Od verze jádra 2.6 je používáno LVM2, které je téměř zcela zpětně kompatibilní s LVM1. V jádře 2.4 se lze setkat s LVM1. Existuje patch, který přináší podporu LVM2 i do jádra 2.4. [25]

Každý PV je rozdělen na menší části, které se označují jako *physical extend* (PE). Tyto bloky mají stejnou velikost jako PE pro VG. Každý LV je rozdělen na menší oblasti, které se označují jako *logical extend* (LE). Bloky mají stejnou velikost pro všechny LV ve VG. [25]

3.3.2 Mapovací režimy LVM

LVM nabízí dvojici základních mapovacích režimů pro mapování LE a PE. [25]

- **Lineární mapování (linear mapping)** - Rozsah PE je přiřazen k prostoru LV v pořadí například: LE 1 - 99 namapován na PV1 a LE 100 - 347 namapováno na PV2.
- **Pruhované mapování (striped mapping)** - Bloky PE jsou prokládány mezi více PV např. *První blok LE[1] -> PV1[1], Druhý blok LE[1] -> PV2[1], Třetí blok LE[1] -> PV3[1], Čtvrtý blok LE[1] -> PV1[2]*

3.3.3 Snímky LVM (Snapshots)

Zajímavou funkcí LVM je tvorba snímků. Tato funkce umožňuje vytvořit nové blokové zařízení, které představuje přesnou kopii LV zmrazeného v určitém časovém okamžiku. Toto zmrazení je prováděno v dávkovém zpracování (např. zálohování), kdy není možné zastavit běžící systém, na kterém se mění data. K snímku lze přistupovat jako k jinému logickému svazku, zatímco původní logický svazek byl již modifikován. [25, 26]

LVM1 má nastaveny snímky jen pro čtení. Snímky pouze pro čtení pracují s tabulkou vyjímky (exception table), která se používá pro sledování bloků, které byly změněny. Je-li blok změněn na původním místě, je nejdříve nakopírován do snímku označeného nejprve jako kopírovaný v tabulce vyjímky a poté jsou zapsána data na původní místo. [25]

V LVM2 je výchozí stav nastaven pro čtení/zápis. Snímek pro čtení/zápis funguje jen pouze pro čtení s další vlastností, že pokud data jsou zapsána do snímku, tak tento blok je označen v tabulce vyjímky jako použitý a nikdy nebude zkopírován podle původního místa. Tímto se otevírá více možností, které nebylo možné vykonat s LVM1 [25]

Důležité je přiřadit dostatek místa pro snímek. Pokud bude snímek LV naplněn, stane se nepoužitelným, a proto je nesmírně důležité vyčlenit dostatek volného místa. Plný snímek je vždy automaticky zablokovan. Množství místa je závislé na využití snímku. Pokud je velikost snímku stejná jako u původního místa, není nutné se obávat přetečení. [25]

3.3.4 Vytvoření LVM

Nejprve je nutné vytvořit *physical volume* (fyzický svazek). Příkaz pro vytvoření je následující. Skládá se z příkazu **pvcreate** a adresy fyzického oddílu, případně i RAID pole (např. */dev/sda*, */dev/md0* apod.). Ověření vytvořeného fyzického svazku je možné provést příkazem **pvdisplay**. [27]

```
$ pvcreate /dev/sda
```

V dalším kroku se vytvoří *volume group* (skupina svazků) a tímto se přiřadí fyzický svazek ke skupině svazků. Příkaz se skládá z **vgcreate** názvu skupiny svazků a adresy fyzického oddílu. Opět je možné zkontrolovat výsledek vytvoření příkazem **vgdisplay**. [27]

```
$ vgcreate vlastni_nazev_vg /dev/sda
```

Poslední částí je vytvoření *logical volume* (logické svazky). Název příkazu je **lvcreate**. Parametr **-L** určuje velikost logického svazku, kde **G** označuje gigabyty. Dále následuje název skupiny svazků. Parametr **-n** přiřazuje název logického svazku. V systému je přístupný logický svazek jako blokové zařízení na adrese */dev/mapper/vlastni_nazev_lv*. Pro kontrolu a výpis lze využít příkazu **lvdisplay**. [27]

```
$ lvcreate -L 6,98G vlastni_nazev_vg -n vlastni_nazev_lv
```

Před samotným použitím je potřeba vytvořit souborový systém a následně připojit do systému.

```
$ mkfs.ext3 /dev/vlastni_nazev_vg/vlastni_nazev_lv  
$ mount /dev/vlastni_nazev_vg/vlastni_nazev_lv /mnt
```

3.4 Iptables

Iptables je firewallová služba, která se stará o zkoumání hlaviček příchozích Transmission Control Protocol/Internet Protocol (TCP/IP) paketů a zabezpečuje práci s nimi. Podle definovaných filtrů je dokáže třídit, přesměrovávat, propouštět nebo zahazovat. Filtrování **iptables** je přímo zabudováno v jádře OS z důvodu výkonu. [28, 29]

Systém **iptables** obsahuje tři základní tabulky. Každá z tabulek je tvořena skupinou řetězců, jenž jsou využívány k daným síťovým účelům. [12, 28, 29]

- **filter** - Tabulka slouží pro filtrování paketů. Obsahuje tři základní řetězce **INPUT**, **OUTPUT**, **FORWARD**. Využívá se pro tvorbu firewallů.

- **nat** - Tabulka slouží pro překlad adres, kdy je přidělena od poskytovatele neveřejná IP adresa. Obsahuje řetězce pravidel kontrolujících Network Address Translation (NAT).
- **mangle** - Tabulka slouží ke zpracovávání hlaviček paketů. Řetězce zpracovávají hlavičky, které následně modifikují nebo zaměňují obsah síťových paketů mimo rámec NAT a filtrování paketů.

Příchozí paket v jádru musí projít některým z řetězů/tunelů (chains). Jsou definovány tři základní řetězce: [28, 29]

- **INPUT** - Všechny přicházející pakety jsou určeny pro daný počítač.
- **OUTPUT** - Všechny pakety které odcházejí z daného počítače.
- **FORWARD** - Všechny procházející pakety, které byly vytvořeny mimo daný počítač, jsou přeposílány (směrovány) jinam. Počítač má funkci routeru.

Řetězec má definovaná pravidla a každý z paketů je zkontrolován všemi pravidly. Je-li úspěšně aplikováno pravidlo, následuje přesun na konec řetězce a provedení dané činnosti. V případě, kdy paket nevyhověl žádnému z nastavených pravidel, je vykonáno hlavní pravidlo tzv. „policy“, které určí, co se stane s paketem. [28, 29]

3.4.1 Vytvoření příkazu iptables

Zadávání příkazů do paketového filtru se provádí příkazem **iptables**. Pro zadávání je nutné mít práva uživatele *root*. [29]

```
$ iptables [tabulka] [akce] [řetězec] [pravidla] [cíl]
```

Není-li specifikována tabulka, je automaticky vybrána tabulka **filter**. V případě, že je potřeba definovat tabulku, lze pomocí parametru **-t** určit název tabulky. Pravidla se starají o samotné filtrování. Za každým pravidlem následuje hodnota definované vlastnosti (např. IP adresa). [29]

Nepostradatelnou částí při definování pravidel je vykřičník. Tento symbol se využívá v případech, kdy je nutné mít význam pravidla opačný (dochází k negaci pravidla). [29, 30]

Tab. 1. Parametry příkazu **iptables** [29]

Parametr	Význam
-A	Na konec řetězce je přidáno nové pravidlo.
-I	Nové pravidlo je přidáno na začátek řetězce.
-D	Smaže pravidlo. Definováno je buď původním zadáním nebo jeho číslem.
-F	Vyprázdnění zadaného řetězce. Všechna pravidla budou smazána.
-L	Vypíše pravidla z daného řetězce.
-P	Zadání hlavního pravidla (policy).
-N	Založení nového řetězce.
-X	Smazání vytvořeného řetězce.
-E	Přejmenování řetězce.
-s	Zdrojová IP adresa paketu, případně rozsah adres (např. 192.168.1.100, 192.168.1.0/255.255.255.0).
-d	Cílová IP adresa paketu nebo rozsahu
-i	Vstupní zařízení, kterým paket připutoval do počítače (např. eth0).
-o	Výstupní zařízení, kudy by měl paket odejít.
-j	Při splnění pravidla se vykoná akce
- -sport	Zdrojový port paketu.
- -dport	Port, na který pakete putuje.

Pravidla jsou schopná rozlišit velkou část paketů. Je-li určeno pravidlo, musí být nastaven i cíl. Určuje, co se stane s paketem. Za parametrem -j následuje název řetězce, který specifikuje akci nebo kam paket spadne. Hlavní akce jsou tři: [29, 30]

- **ACCEPT** - Paket je akceptován, filter jej propouští dále.
- **DENY, REJECT** - Paket je zahozen. O zahození je informován zdrojový počítač.
- **DROP** - Paket je zahozen. Zdrojový počítač není informován.
- **LOG** - O existenci paketu je proveden záznam do logu pomocí **syslog**. Neurčuje co se stane s paketem.

3.5 Cron

V Linuxu se o pravidelné spuštění příkazů stará daemon **cron**. Daemon je vždy spuštěn se startem operačního systému a běží po celou dobu chodu systému a pravidelně čte konfigurační soubor. Ten soubor má zapsán časy a seznamy příkazové řádky, které vyvolává shell. Co lze spustit ručně z shellu, je možné spustit i v **cronu**. [12]

3.5.1 Crontab

Záznamy pro vykonání **cronem** jsou uloženy v systémových a uživatelských tabulkách. Tyto tabulky jsou nazývány **crontaby**. K systémové tabulce má přístup pouze uživatel *root*. Ke zbývajícím tabulkám mají přístup samotní uživatelé systému. Z těchto tabulek daemon zjišťuje, které programy nebo příkazy mají být spuštěny a kdy. Tabulky mohou být uloženy v konfiguračních souborech na třech různých místech. Při spuštění hledá **cron** v adresáři */var/spool/cron* (v systému Debian je to */var/spool/cron/crontabs*) soubory pojmenované jako uživatelské účty. Dále hledá v adresáři */etc/cron.d/* a */etc/crontab*. [1, 12]

Každou minutu hledá pak všechny takto získané údaje, které kontroluje, jestli nemá být právě něco spuštěno a pak usne. Programy jsou spouštěny jako uživatel vlastníci **crontab**. Pokud dojde k zmeškání úlohy z důvodu vypnutí systému nebo nesprávného systémového času a jeho následující aktualizací, je příslušná úloha nevykonána. **Cron** je velmi citlivý na čas. [1, 12]

3.5.2 Nastavení crontabu

Nastavení **crontabu** je možné z příkazové řádky zadáním příkazu:

```
$ crontab -e
```

Proběhne spuštění textového editoru, ve kterém je možné zadat potřebné údaje. První pozice je minuta (rozsah 0 - 59), následuje hodina (0 - 23), den (1 - 31), měsíc (1 - 12) a den v týdnu (0 - 6, kdy 0 označuje neděli).

Jednotlivé položky mohou být zadány následujícími způsoby: [12]

- Hvězdička „*“ určuje, že zadání vyhovuje všem časům.
- Celé číslo (např. 12) určuje přesný čas zadání.
- Dvě čísla oddělená pomlčkou (např. 1-12) určují, že zadání vyhovuje rozsahu zadaných hodnot.

- Čísla oddělená mezi sebou (např. 1,5,9) určují, že zadání vyhovuje zadaným hodnotám.

```
#m h dom mon dow  command
0 11 * * * (jmeno uživatele) nazev_prikazu
0 10 * * * (jmeno uživatele) nazev_prikazu
```

Pro vypsání jednotlivých položek **crontabu** je použitý příkaz:

```
$ crontab -l
```

3.6 Syslog - logování systémových událostí

Je to logovací systém pro zaznamenávání událostí, kdy systém hlásí různé chyby, varování a jiná důležitá hlášení. O logování systémových událostí se v Debianu stará **rsyslog**. Zabezpečuje přístup ke zprávám, které je možné prohlížet později nebo s velkým časovým odstupem. Zprávy mohou být uloženy na lokálním počítači nebo mohou být směřovány na jiný počítač v síti, který je nazýván syslog server. [1, 12]

Rsyslogd je logovací daemon a třídí tyto zprávy a hlášení do souborů podle důležitosti. Spouští se vždy spouštěním operačního systému a běží po celou dobu běhu počítače. Daemon zajišťuje nahlížení do konfiguračního souboru a odeslání zprávy na určené místo. [1, 12]

3.6.1 Nastavení rsyslogd

Konfigurace **rsyslogd** se provede v konfiguračním souboru */etc/rsyslog.conf*. Jedná se o textový soubor. Obsahem jsou příkazy pro chování daemona. Formát příkazu je: [12]

```
selektor <tab> akce
```

Selektor určuje program (službu), který odesílá zprávu do log souboru. Za selektorem je určena úroveň (syntax *služba.úroveň*). Selektor může obsahovat více služeb, jenž jsou mezi sebou odděleny čárkami. V případě, kdy je potřeba výběru všeho nebo ničeho, lze dosadit za selektor symbol hvězdičky „*“ (výběr všeho) nebo klíčového slova *none* (výběr ničeho). Příklad několika způsobů zápisu: [12]

```
služba.úroveň                akce
služba1,služba2.úroveň      akce
služba1.úroveň1; služba2.úroveň2;  akce
*.úroveň                    akce
*.úroveň; špatnáúroveň.none    akce
```

Úroveň označuje prioritu důležitosti zprávy. V konfiguračním souboru je úroveň udávána jako minimální priorita. Určuje, kdy bude uložena do log souboru. Za akci se dosazuje vykonání záznamu zprávy. Zpráva může být buď uložena na lokálním počítači nebo může být odeslána na logovací server, kde je uložena. [12]

4 TVORBA ZÁLOHOVACÍHO SKRIPTU

4.1 Zásady tvorby skriptů

Při tvorbě skriptu je vhodné dodržet určitá pravidla pro jejich tvorbu. Pro tvorbu skriptů lze využít libovolného textového editoru (**vim**, **gedit**, **nano**, **kate**). Vhodné je i to, pokud umí editor zvýraznit syntaxi, která pak činí psaní skriptů přehlednějším. Skript by měl být pojmenován podle vykonávané činnosti. Důležité je i ověření, zda zvolený název nekoliduje s již existujícími příkazy. [1]

Vlastnosti dobrého skriptu by měly splňovat obecná doporučení: [1]

- Skript má běžet bez chyb.
- Skript má dělat to, k čemu je určen.
- Logika programu má být jasně definovaná a zřejmá.
- Skript nemá dělat nic zbytečného.
- Skript má být použitelný univerzálně.

4.2 Příkazový interpret Bourne Again Shell (BASH)

Shell, označován také jako příkazový interpret, je rozhraní, které interpretuje uživatelské příkazy zadané uživatelem do linuxového systému. Příkazy mohou být zadány buď uživatelem a nebo načítané ze souboru, který se nazývá shellový skript. Shellové skripty jsou příkazovým interpretem interpretovány, tedy nejsou kompilovány. Shell čte skript řádek po řádku a hledá v systému příkazy k provedení. Hlavním úkolem shellu je také poskytnutí uživatelského rozhraní, které lze nastavit v konfiguračních souborech. [1, 10] Linux má vlastnost modularity, a proto je možnost využití jiného libovolného shellu. Od původního Bourne shellu je odvozena většina jiných shellů. Standardním shellem je BASH **/bin/sh**, který je vždy nainstalován na všech linuxových systémech a patří do rodiny nástrojů GNU. Shell je přenositelný na všechny varianty Unixu, protože splňuje normu POSIX. Výchozím shellem ve většině distribucí je **/bin/sh**, který odkazuje na **/bin/bash**. BASH poskytuje možnost editace příkazové řádky, historie příkazů s neomezenou délkou, řízení úloh, funkce a aliasy, indexovaná pole s neomezenou velikostí

a celočíselnou aritmetiku se základem 2 až 64. Pomocí kompatibility může být skript pro Bourne shell spuštěn BASHem. [1, 10, 11]

4.2.1 Příkazy pro BASH

Shell obsahuje vestavěné příkazy. Je-li uvedeno jako první slovo název vestavěného příkazu, provede shell tento příkaz přímo a nebude vytvořen nový proces. Vestavěné příkazy jsou důležité k zajištění funkcí, které by bylo nemožné či nepohodlné realizovat samotnými programy. BASH podporuje tři typy vestavěných příkazů. [1]

- Vestavěné příkazy Bourne shellu: **:, ., break, cd, continue, eval, exec, exit, export, getopts, hash, pwd, readonly, return, set, shift, test, [, times, trap, umask, unset.**
- Vestavěné příkazy BASHe: **alias, bind, builtin, command, declare, echo, enable, help, let, local, logout, printf, read, shopt, type, typeset, unlimited, unalias.**
- Speciální vestavěné příkazy: Je-li BASH spuštěn v režimu POSIX, liší se chování speciálních vestavěných příkazů od ostatních vestavěných příkazů v následujících bodech:
 - Názvy speciálních vestavěných příkazů se detekují dříve než názvy funkcí shellu.
 - Skončí-li speciální vestavěný příkaz chybou, neinteraktivní shell bude ukončen. Příkazy přiřazení předcházejí příkazu a zůstanou po skončení příkazu v platnosti.

Speciální POSIXové vestavěné příkazy jsou **:, ., brek, continue, eval, exec, exit, export, readonly, return, set, shift, trap, unset.**

BASH nabízí pro rozvětvení a zvýšení účinnosti podmíněné příkazy, regulární výrazy, funkce a opakovatelné operace, které lze využít ve skriptech.

4.2.2 Spuštění shellového skriptu

Skript lze spustit dvěma způsoby. Základní spuštění je založeno na vyvolání, kterému předáme název skriptu v parametru. [10]

```
$ /bin/sh nazev_skriptu
```

Příkaz bude proveden, ale pro jednodušší spuštění je vhodnější zadat jen název skriptu stejně jako u libovolného jiného linuxového příkazu. Pomocí příkazu **chmod** se změjí práva souboru pro všechny uživatele na spustitelný. [10]

```
$ chmod +x nazev_skriptu
```

Nyní lze skript spustit příkazem: [10]

```
$ nazev_skriptu
```

Může se stát, že se objeví chyba s oznámením, že příkaz nebyl nalezen. Příčinou je shellová proměnná *PATH*, která nevyhledává spustitelné příkazy v aktuálním adresáři. Musí být definována buďto na příkazovém řádku zápisem *PATH = \$PATH:.*, nebo v editoru otevřít skript *.bash_profile* a zapsat stejný příkaz nakonec; potom se odhlásit ze systému a znovu přihlásit. [10]

Doporučeným způsobem je spouštění skriptu zápisem *./*, které je před názvem skriptu. Při spuštění se nemůže stát, že by se spustil nahodile v systému jiný příkaz, který má stejný název. [10]

```
$ ./nazev_skriptu
```

Při spuštění shellového skriptu se vytvoří *BASH fork* (potomka) nového procesu *BASHe*. Tento subshell čte ze skriptu řádek po řádku. Načtený řádek je přečten, interpretován a proveden příkaz tak, jako by byl zadán z klávesnice. [1]

Subshell vykonává jednotlivé řádky a jeho otcovský shell čeká na ukončení synovského procesu. Po zpracování celého skriptu se subshell ukončí. Pak je probuzen rodičovský shell, který zobrazí *prompt*. [1]

4.3 Programovací jazyk GNU *awk*

Gawk je GNU verze známého unixového programu **awk**⁷⁾. Program **awk** je velmi často odkazem na **gawk**. Základní funkcí **awk** je vyhledávat v souboru řádky nebo jiné textové řetězce, které obsahují hledaný vzor. Pokud řádek vyhovuje vzoru, pak bude provedena požadovaná operace. [1]

Programy v **awku** jsou daty ovládané programy. To znamená, že popisují data s kterými se pracuje a co se s nimi chce dělat. Programy vytvořené v **awku** se snadno vytvářejí i čtou. [1]

Pomocí nástroje **awk** lze provádět jednoduché přeformátování textu. Program čte řádky vstupního souboru a rozeznává jednotlivé vstupní sloupce. Za běhu je možnost definovat

⁷⁾Jazyk AWK vytvořila v 70. letech trojice programátorů - Aho, Kernighan a Weinberger. Název je tvořen prvními písmeny jejich jmen. [1]

proměnné a snadno počítat součty, statistiky a další údaje nad vstupními daty. Příkazy a proměnné je možné definovat i ve skriptech. [1]

4.3.1 Příkazy pro `awk`

Existuje několik možností, jak `awk` spouštět. Pro krátký program je vhodné spouštění z příkazové řádky. [1]

```
awk PROGRAM vstupní_soubor(y)
```

Pokud jsou prováděny složitější změny pravidelně a nad více soubory, je vhodné zadat příkazy `awk` ve skriptu. Výsledný zápis pro spuštění je následující: [1]

```
awk -f SOUBOR_S_PROGRAMEM vstupní_soubor(y)
```

4.4 Editor Stream Editor (`sed`)

Editor `sed` umožňuje základní transformaci textu načítaného ze souboru nebo roury (znak `|`). Výsledek transformace je zaslán na standardní vstup. Příkaz `sedu` nedovoluje zadat výstupní soubor. Výsledek je možné uložit přes přesměrování do souboru. Při transformaci textu nedochází ke změně původního souboru. [1]

Editor `sed` při běhu nekomunikuje s uživatelem. Díky této vlastnosti je nazýván jako *dávkový editor*. Tyto vlastnosti ho předurčují k editačním účelům ve skriptech a umožňuje opakovanou editaci textových souborů. [1]

4.4.1 Příkazy pro `sed`

Editor `sed` zvládá nahrazovat a mazat text podle předem určeného vzoru. K tomu využívá regulárních výrazů, které využívá program `grep`. Editační příkazy jsou obdobné jako u editoru `vi`. [1]

Mimo příkazů pro editaci je možné zadat i parametry. [1]

Tab. 2. Příkazy editoru sed [1]

Příkaz	Význam
a\ c\ d i\ p r s w	Přidá text za aktuální řádek. Nahradí text na aktuálním řádku. Vymaže text. Vloží text před aktuální řádek. Vytiskne text. Čte soubor. Hledá a nahrazuje text. Zapisuje do souboru.

Tab. 3. Volby příkazu sed [1]

Volba	Význam
-e skript	Přidá <i>skript</i> příkazům, které budou prováděny při zpracování vstupu.
-f soubor	Přidá příkazy v <i>souboru</i> k příkazům, které budou prováděny zpracování vstupu.
-n	Tichý režim.
-V	Vypíše verzi programu a skončí.

4.5 Secure Shell (SSH)

SSH byl původně vyvinut studentem Tatu Ylonenem na univerzitě v Helskách. Specifikace protokolu a jeho balíček jsou bezpečnou náhradou k programům pro vzdálené relace a přenos souborů **telnet**, **rnp** a **rlogin**. Protokol SSH podporuje zašifrovanou komunikaci mezi účastníky a také ověření vzdáleného hostitele. Minimalizuje situace jako je napodobení klienta pomocí falšování IP adresy, útoku „man-in-the-middle“ nebo manipulaci s Domain Name System (DNS). Ověření vzdáleného hostitele je zajištěno více šifrovacími protokoly, které využívají různé druhy tajných klíčů (DES, 3DES, IDEA, Blowfish). Tímto je zajištěna důvěrnost komunikace již od počáteční výměny uživatelského jména a hesla. Komunikace mezi klientem a vzdáleným zařízením je šifrována. Umožňuje kompresi dat a zabezpečenou X Window komunikaci. Bezpečnost a důvěryhodnost zvyšuje i to, že protokol je standardizován organizací Internet Engineering Task Force (IETF). [1, 12, 30]

Dnes existuje více implementací SSH. Původní komerční verzi vyvinula firma Data Fellows. Z původního SSH vznikla open source varianta. Tato verze neobsahuje patentové ani proprietární části. Jméno projektu je OpenSSH a v současnosti je ve verzi 5.8 (duben 2011). Důležité části jsou serverový daemon **sshd**, **ssh** pro zadávání uživatelských příkazů, vzdálené přihlášení a dále **scp** pro zabezpečené kopírování. Neméně důležitou částí je i program **ssh-keygen** pro generování páru klíčů (privátního a veře-

jného) a další nástroje pro podporu bezpečného spojení s X Window. OpenSSH pracuje s SSH verze 1 a SSH verze 2. OpenSSH lze získat na adrese <http://www.openssh.com>. [1, 12, 30]

4.5.1 Metody ověření identity

V souboru `/etc/ssh/ssh_config` jsou volby, které jsou spojené s autentizací uživatele. SSH umí ověřit identitu několika způsoby: [12]

- **Metoda A** - Je-li jméno počítače, ze kterého se uživatel přihlašuje uloženo v souborech `/.rhosts`, `/.shost`, `/etc/hosts.equiv` nebo `/etc/shosts.equiv`. Uživatel je ihned přihlášen do systému bez zadání hesla. Tento způsob je stejného typu jako u programu **rlogin** a není moc vhodný pro normální běh systému.
- **Metoda B** - **sshd** je použito pro ověření identity vzdáleného počítače šifrovací metodou veřejných klíčů. Veřejný klíč vzdáleného zařízení je uložen v souboru `/etc/ssh/known_hosts` nebo v souboru `/.ssh/known_hosts` v domácím adresáři. Uživatelův počítač se musí prokázat odpovídajícím privátním klíčem. Po úspěšné autentizaci je uživatel přihlášen do systému. Metoda není bezpečná, protože v případě napadení vzdáleného zařízení je v ohrožení i uživatelův počítač.
- **Metoda C** - **sshd** je použito pro ověření identity vzdáleného počítače šifrovací metodou veřejných klíčů. Při přihlášení musí uživatel disponovat svým privátním klíčem a musí poskytnout heslo k dešifrování. Tato metoda je nejbezpečnější.
- **Metoda D** - **sshd** umožňuje uživateli zadat heslo. Z **ssh** se stavá jakoby klasický **telnet**, ale celá komunikace je šifrována. Nevýhodou je prolomení uživatelského hesla.

4.5.2 Man-in-the-middle (muž uprostřed)

Jedná se o druh útoku, kdy je útočník mezi uživatelem a cílovým serverem. Útočník se vydává za cílový server a odchyťává komunikaci nebo hesla. Komunikaci pak může předávat na cílový server. Tento typ útoku se využívá při ověření IP adresy nebo DNS záznamu. Jediným bezpečným způsobem je využití autentizace pomocí SSH. [28]

4.5.3 Připojení na vzdálené zařízení pomocí ověření klíčů

Pro generování páru klíčů se využívá příkazu **ssh-keygen**. Program zajistí vytvoření veřejného klíče (většinou v $\$HOME^8$)/*.ssh/id_rsa.pub*) a privátního klíče (většinou v $\$HOME$)/*.ssh/id_rsa*, je chráněn heslem). Klíčový pár SSH je založen na asymetrickém šifrování. Veřejný klíč je nutné před prvním použitím nahrát na vzdálené zařízení do souboru *.ssh/authorized_keys* v domovském adresáři. Poté se již lze přihlásit pomocí metody ověření klíčů. Klient a server provedou mezi sebou kryptografické ověření shody obou klíčů. Při připojení uživatele vyše server posloupnost dat (server žádá ověření identity). Uživatel přijatá data zašifruje svým privátním klíčem a odešle je zpět serveru. Ten je dešifruje svým veřejným klíčem a v případě shody je uživatel přihlášen do systému. [31, 32, 33]

Vhodným bezpečnostním opatřením je ochrana privátního klíče. V případě jeho zcizení se může útočník připojit na všechna vzdálená zařízení, kde je umístěn veřejný klíč (tedy v souboru *authorized_keys*). [31]

4.6 Domain Name System (DNS)

Domain Name System (DNS) slouží k překladu doménových jmen (domena.cz) na IP adresy (192.168.2.20) a naopak. Tato možnost byla vytvořena proto, že uživatelé si lépe pamatují jménou adresu než číselnou kombinaci. DNS serverů existuje po celém světě velké množství. Systém serverů je sestaven do podoby velkého stromu. Je to z toho důvodu, aby si každý DNS nemusel pamatovat informace o všech doménách. Každý ze serverů zná část své domény směrem vzhůru. Nejvyšší DNS servery se starají o kořenovou doménu (např. *.cz*) a je nadražena všem doménám. [28, 29]

4.6.1 DNS zone transfer

DNS zone transfer zajišťuje přenos celé doménové zóny (jejich záznamů) z jednoho DNS serveru na druhý. Tento přenos je použit u přenosu mezi primárním a sekundárním DNS v případě změny obsahu zóny. Přenos může být proveden dvěma metodami. Základním metodou je AXFR, která přenáší celý obsah domény i v případě, kdy došlo jen ke změně jednoho záznamu. Aby se nemusela přenášet celá zóna, byla zavedena metoda IXFR, která přenáší jen změny u obsahu zóny. [34, 35]

Dotazy na obsah zóny přes AXFR by neměl být veřejně přístupný. Pokud by se tak stalo, mohl by si každý zájemce stáhnout veškeré záznamy domény. Tímto by se zájemce mohl dostat k seznamu subdomén v zóně, seznamu počítačů apod. Dotazy by měly být povoleny jen pro ověřené DNS servery, které mezi sebou musí komunikovat. [34]

⁸⁾ $\$HOME$ značí umístění pro domovskou složku tedy např. */home/uživatel*

Příkaz **host** je jednoduchý program pro provádění DNS dotazů. Díky němu lze získat výpis z doménové zóny z DNS, který lze dále zpracovat podle vlastních požadavků.

```
$ host -l domena.cz
prvni.domena.cz has address 192.168.1.1
druha.domena.cz has address 192.168.1.2
treti.domena.cz has address 192.168.1.3
```

Příkaz **host -l .domena.cz** by neměl být přístupný z veřejné sítě. V případě zálohovacího skriptu je přístup zajištěn jen z vnitřní sítě.

II. PROJEKTOVÁ ČÁST

5 DŮVODY VYTVOŘENÍ ZÁLOHOVACÍHO SERVERU

Podnětem pro zahájení činnosti zálohovacího serveru bylo vytvořit centralizované místo pro shromažďování provozních log souborů z různých vzdálených zařízení, které jsou rozmístěny v různých lokalitách. Součástí serveru je **syslog** daemon pro zaznamenávání systémových událostí. V případě výpadku některého ze vzdálených zařízení jsou provozní log souboru umístěny na zmíněném zálohovacím serveru. Výhodou této centralizace je snadné dohledávání potřebných údajů pro správu jednotlivých zařízení.

Pravidelné zálohování log souborů se provádí každý den ve stanovenou dobu. Tyto soubory jsou rozděleny podle určených kritérií. Jako je název (adresa) vzdáleného zařízení, rok a měsíc. Každý soubor je patřičně pojmenován pro případné vyhledání. Soubory starší než 180 dnů jsou z důvodu kapacity a přehlednosti mazány. Kopírování log souborů se provádí pomocí SSH a **scp**.

Zálohování se provádí na oddíl, který je umístěn v RAID-1 s vytvořeným oddílem LVM2. V případě potřeby přidání dalšího místa lze oddíl rozšířit za provozu. Operační systém je zabezpečen pomocí **iptables**. Do systému je možné se přihlásit z určitých IP adres a provozovat pouze povolené služby na určených portech, které projdou přes firewall.

Popisovaný způsob zálohování využívá jeden poskytovatel internetového připojení ve Zlínském kraji, pro kterého byla vytvořena tato bakalářská práce.

6 INSTALACE OPERAČNÍHO SYSTÉMU

Instalovaným operačním systémem je GNU/Linux Debian 6.0.1 Squeeze. Instalační obraz byl vybrán o velikosti 40MB (tzv. business card). Z obrazu jsou nataženy pouze potřebné data pro zavedení instalace. Zbytek potřebných dat při instalaci je stahováno přímo z Internetu.

Při instalaci byli vytvořeni dva uživatelé (*root* - práva superuživatele, *thomas* - práva uživatele). Server je pojmenován *pepina* a byl přiřazen do příslušné domény.

V této části je popisován průběh instalace. Popisovány jsou pouze důležité body, které se liší od standardní instalace. Veškerá jména a IP adresy jsou smyšlené z důvodu bezpečnosti serveru.

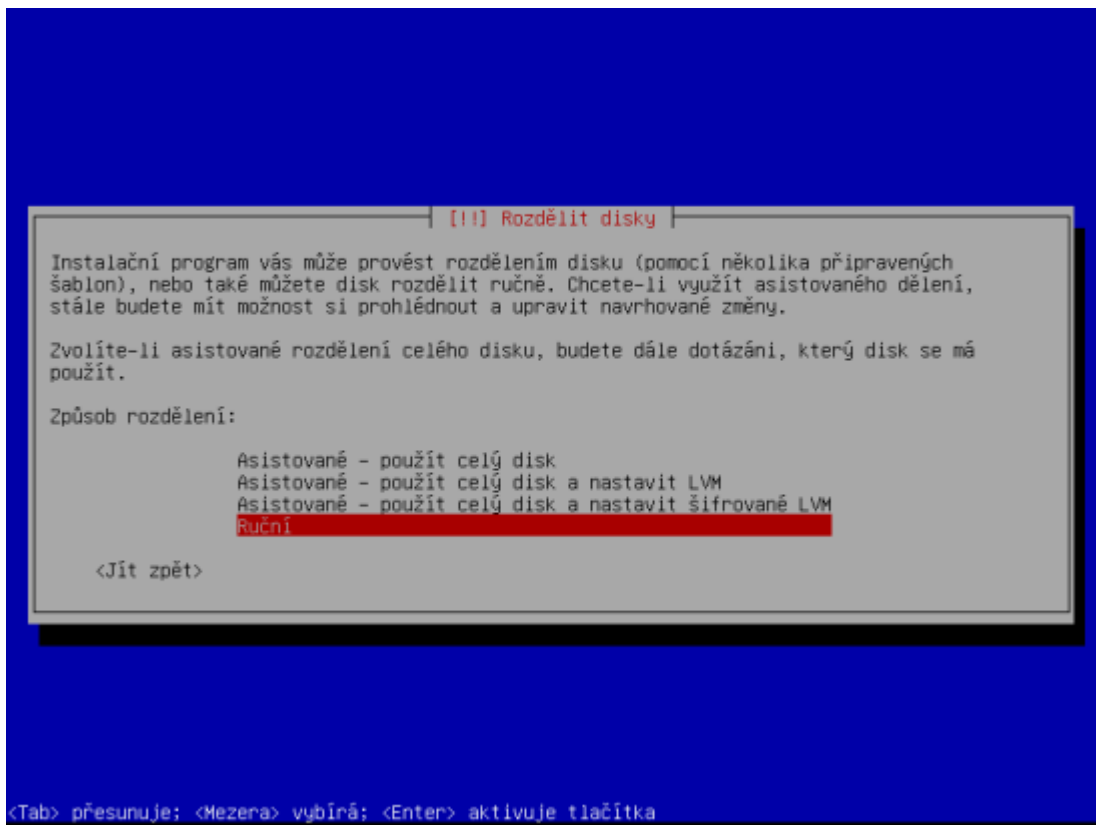
6.1 Vytvoření RAID pole

V průběhu instalace je vytvořen softwarový RAID-1 přímo z instalátoru Debianu. Server má dva disky. Každý z disků je rozdělen následovně:

- Oddíl pro systém

- Oddíl pro odkládací prostor (swap)
- Oddíl pro data

V instalátoru při rozdělování disků je nutné zvolit položku **Ruční**.

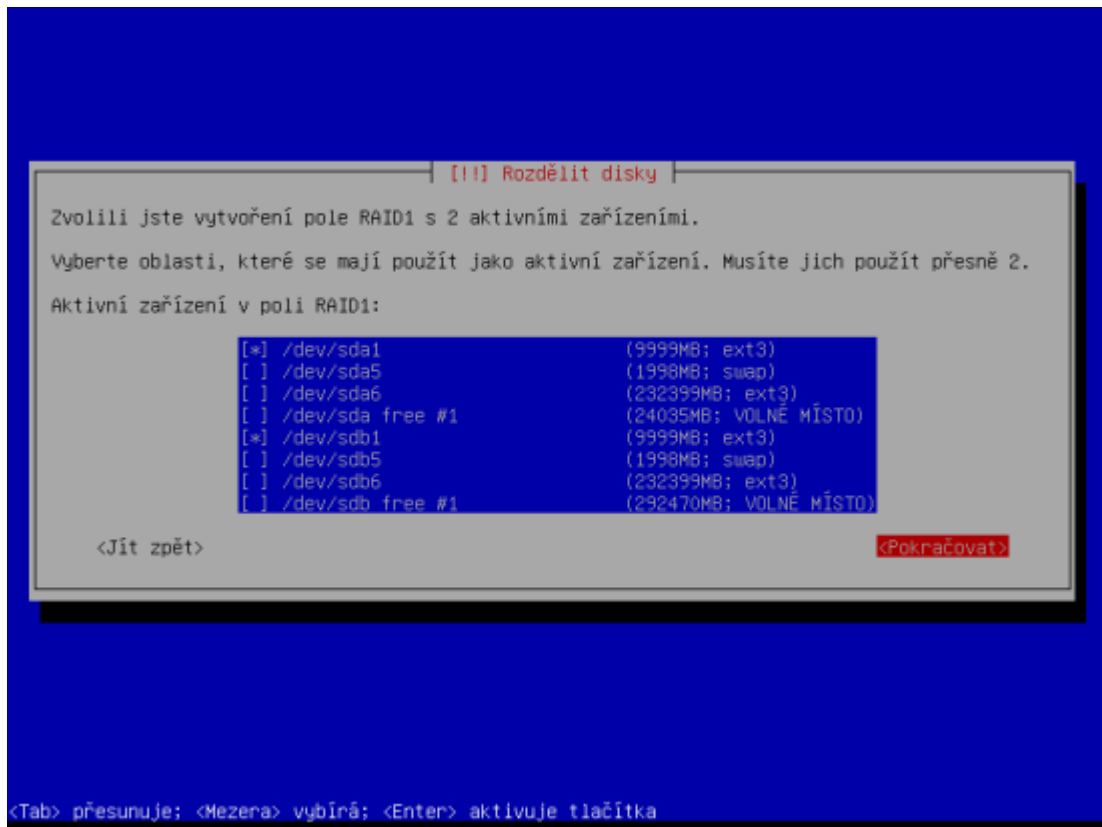


Obr. 10. Volba asistovaného dělení disku.

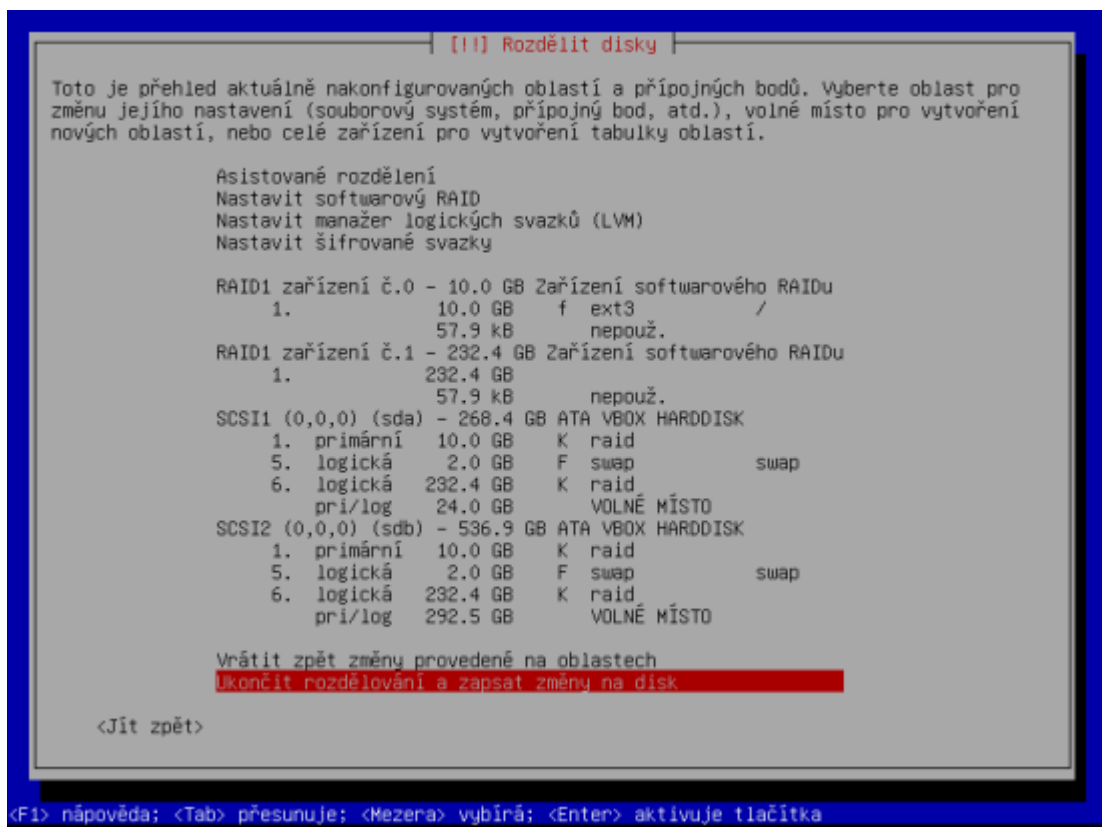
Na každém disku se musí vytvořit nová tabulka oblastí. Poté je možné vytvořit jednotlivé oddíly podle zmíněného rozdělení. Pro systémový oddíl je zvolena primární oblast. Odkládací oddíl a data jsou v logické oblasti.

Výběrem volby **Nastavit softwarový RAID** se objeví výzva pro zformátování vytvořených oddílů. Poté se vybere volba **Vytvořit MD zařízení** a následně je nutné určit typ pole RAID. Pro toto řešení bylo vybráno pole typu RAID-1. V následujících krocích se zadá počet aktivních zařízení v poli a počet rezervních zařízení. Počet aktivních zařízení je dva (`/dev/sda` a `/dev/sdb`). Pole nemá žádné rezervní zařízení, a proto bude vložena nula. V dalším kroku je nutné vybrat oddíly, které budou součástí RAID pole. Po výběru příslušných oddílů jsou zapsány změny a nastavení RAIDu.

Odkládací prostor je formátován jako swap. Ze systémových oddílů se vytvoří RAID-1 jako `/dev/md0` a z oddílu pro data je opět RAID-1 označený jako `/dev/md1`. Pole `/dev/md0` je zformátováno pod ext3. RAID-1 má dva aktivní disky a žádné rezervní zařízení. Nad `/dev/md1` se vytvoří LVM2 oddíl po instalaci systému.



Obr. 11. Výběr oddílů do RAID pole.



Obr. 12. Rozdělení disků na oddíly a RAID pole.

6.2 Vytvoření LVM2

Vytvoření LVM2 je provedeno až po samotné instalaci operačního systému. Veškeré systémové úpravy jsou prováděny pod uživatelem *root*. Tento uživatel má superpráva pro správu celého systému a jakákoliv chyba může vést k poškození chodu systému.

Balík pro vytváření a správu LVM2 se nenachází v základní instalaci, a proto je nutné ho nejprve nainstalovat. Instalace se provede příkazem:

```
# aptitude install lvm2
```

Nový LVM2 oddíl se vytvoří nad polem RAID označeném jako */dev/md1*, kde budou uložena uživatelská data. Prvním krokem je vytvoření *physical volume* (fyzického oddílu). *Physical volume* se vytvoří příkazem:

```
# pvcreate /dev/md1
```

Ověření vytvoření a dodatečné informace o novém *physical volume* je možné získat příkazem:

```
# pvdisplay
```

Následně se vytvoří *volume group* (skupina svazků). Název pro *volume group* je zvolen *zaloha_logy*. Příkaz pro vytvoření je:

```
# vgcreate zaloha_logy /dev/md1
```

Opět je možné si ověřit a získat informace o nově vytvořené *volume group* příkazem:

```
# vgdisplay
```

Posledním zbývajícím krokem je vytvořit *logical volume* (logický svazek). Do příkazu vytvoření *logical volume* se vloží jeho velikost, název *volume group* a název pro *logical volume*. Název nového *logical volume* je *zalohovani*. Příkaz má následující syntax:

```
# lvcreate -L 160G zaloha_logy -n zalohovani
```

I zde je možnost ověřit úspěšné vytvoření *logical volume* příkazem:

```
# lvdisplay
```

Před samotným použitím se musí nový *logical volume* zformátovat. Nový LVM2 oddíl je zformátován na ext3. Příkaz pro zformátování je:

```
# mkfs.ext3 /dev/zaloha_logy/zalohovani
```

Aby bylo možné pracovat s LVM2 oddílem je potřeba ho připojit do systému. Připojení lze provést příkazem:

```
# mount /dev/zaloha_logy/zalohovani /mnt
```

V případě automatického připojení (po startu operačního systému) se musí upravit soubor (tabulka diskových oddílů) uložený v */etc/fstab*. Následující řetězec znaků se vložit do souboru *fstab*.

```
/dev/zaloha_logy/zalohovani /mnt ext3 auto,user 0 0
```

7 ZABEZPEČENÍ OPERAČNÍHO SYSTÉMU

7.1 Zabezpečení pomocí iptables

Zabezpečení operačního systému proti nechtěným návštěvníkům a vpuštění jen povolených služeb a uživatelů zajišťují pravidla **iptables**. Pravidla jsou definována ve skriptu **firewall.sh**, který je uložen v */etc/init.d* a skript je ihned po startu spuštěn s nastavenými pravidly.

Firewallový skript se skládá ze dvou funkcí:

- **start_firewall** - Zabezpečuje vložení potřebných pravidel, které mají být nastaveny. Tento stav vyjadřuje stav běžící.
- **stop_firewall** - Korektně ukončí běh nastavených pravidel. Všechna nastavená pravidla jsou zrušena a pravidla **iptables** jsou nastavena na výchozí hodnoty.

V hlavičce firewallového skriptu se nachází deklarace *LSBInitScripts*, zajišťující spuštění skriptu v */etc/init.d* po startu nebo případné zastavení či restartování firewallu. Deklarace *LSBInitScripts* je výchozím nastavením v Debianu Squeeze.

Po deklaraci hlavičky skriptu následuje definování povolených IP adres, ze kterých je možno se připojit na zálohovací server. Restriktivní povolení je i pro služby, které server nabízí. Proto, aby bylo spojení úspěšné, je nutné mít povolenou IP adresu pro přístup a připojovat se na službu s povoleným portem, na kterém služba naslouchá. V případě tohoto zálohovacího serveru je povolen přístup pro definované IP adresy, služby **SSH**, **rsyslog** a jejich porty.

Funkce **start_firewall** má pevně definována základní pravidla (tzv. policy) pro tabulku **filter**. Všechny příchozí pakety jsou zahazovány. O zahození není informován zdrojový počítač. Všechny odchozí pakety jsou propouštěny dále. Ve skriptu následuje smyčka cyklů pro povolení jednotlivých služeb a jejich portů pro IP adresy, které využívají spojení typu TCP v příchozím směru. Obdobná smyčka cyklů je pro služby a povolené IP adresy používající spojení typu User Datagram Protocol (UDP).

Na zálohovací server jsou povoleny příchozí pakety typu Internet Control Message Protocol (ICMP) ze všech počítačů z důvodu zjištění dostupnosti zálohovacího serveru a jeho služeb. Povoleny jsou příchozí pakety v rámci localhostu (loopback). Pakety přicházející na vnější rozhraní (eth0) typu TCP, které nejsou povoleny, jsou zahozeny. Je předán RST příznak do paketů a následně je spojení ukončeno. V případě spojení UDP a paketů přicházejících na vnější rozhraní u kterých není povoleno spojení, jsou pakety zahozeny a je oznámeno zdrojovému počítači hlášení *ICMP Port Unreachable*. Funkce **stop_firewall** má nastavená pravidla do výchozího stavu. To znamená, že všechny příchozí, odchozí pakety jsou přijaty a může probíhat komunikace. Při spuštění těchto pravidel jsou předchozí pravidla z funkce **start_firewall** zrušena.

Poslední částí firewallového skriptu je rozhraní pro spouštění přes příkazovou řádku. Firewall lze spouštět ve třech stavech:

- **Spuštění** - Odpovídá popisu funkce **start_firewall**. Spustit firewall lze příkazem **firewall.sh start**.
- **Zastavení** - Odpovídá popisu funkce **stop_firewall**. Spustit firewall lze příkazem **firewall.sh stop**.
- **Restart** - Restartováním se zajistí zastavení a znovu spuštění firewallu (funkce **stop_firewall** je spuštěna a následně je spuštěna funkce **start_firewall**). Akci je možné provést přes příkaz **firewall.sh restart**

Firewallový skript se spouští ze složky */etc/init.d*. Pro zajištění spouštění po startu počítače je potřeba vytvořit symbolický odkaz na firewallový skript do složek, které reprezentují jednotlivé úrovně běhu systému. Jedná se o úrovně, jenž jsou reprezentovány složkami */etc/rc0.d*, */etc/rc1.d*, */etc/rc2.d*, */etc/rc3.d*, */etc/rc4.d*, */etc/rc5.d* a */etc/rc6.d*. Písmeno S určuje spouštění služby a K značí ukončení.

Příkazy pro vytvoření symbolických odkazů jsou:

```
# ln -s /etc/init.d/firewall.sh /etc/rc0.d/K98firewall.sh
# ln -s /etc/init.d/firewall.sh /etc/rc1.d/K98firewall.sh
# ln -s /etc/init.d/firewall.sh /etc/rc2.d/S98firewall.sh
# ln -s /etc/init.d/firewall.sh /etc/rc3.d/S98firewall.sh
```

```
# ln -s /etc/init.d/firewall.sh /etc/rc4.d/S98firewall.sh
# ln -s /etc/init.d/firewall.sh /etc/rc5.d/S98firewall.sh
# ln -s /etc/init.d/firewall.sh /etc/rc6.d/K98firewall.sh
```

Po vytvoření symbolických odkazů pro jednotlivé úrovně je ještě potřeba zaktualizovat se seznamem služeb v **update-rc.d**. Provede se to příkazem:

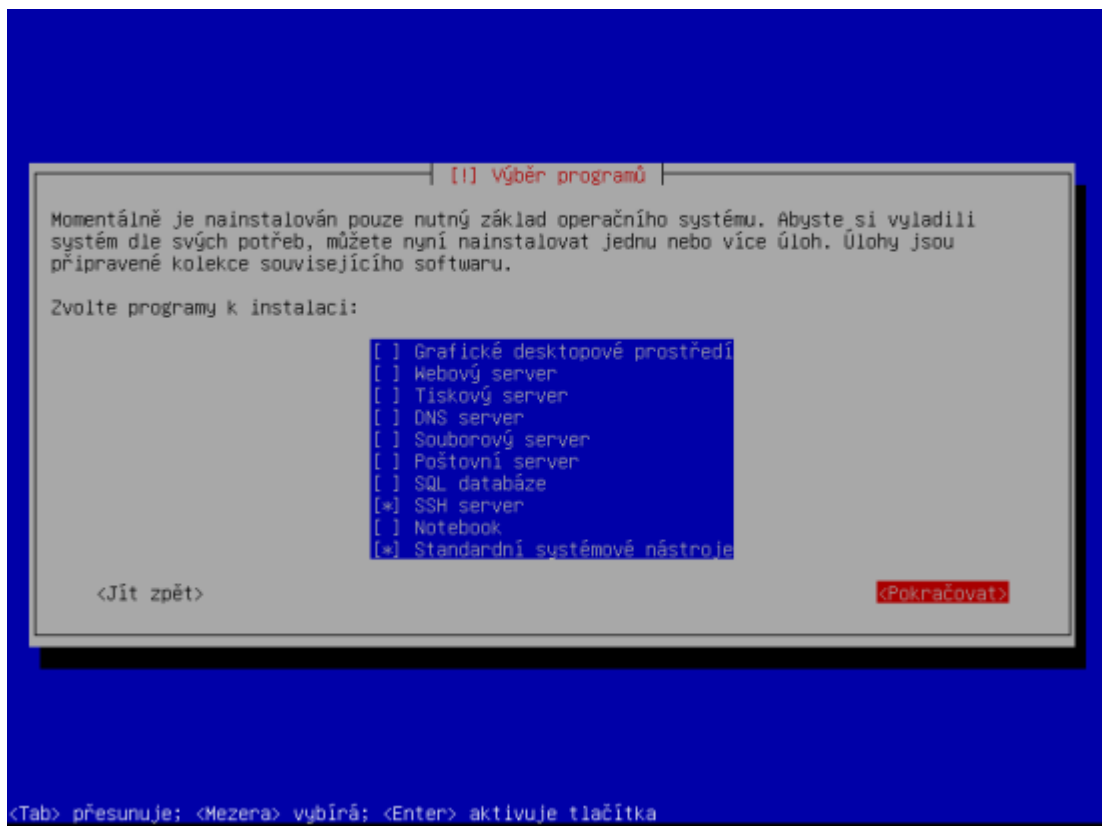
```
# update-rc.d firewall.sh defaults
```

Kontrola nastavených pravidel se zjistí příkazem:

```
# iptables -L
```

7.2 Zprovoznění SSH serveru

Aby bylo možné server vzdáleně spravovat, musí se nainstalovat balík *openssh-server*, který zajišťuje na serveru to, že je možné se na něj přihlásit. Tento balík lze nainstalovat buď při instalaci samotného operačního systému, kdy se vybere volba **SSH server**.



Obr. 13. Instalace SSH serveru z instalátoru Debianu

Případně je možné balík nainstalovat po instalaci systému. Příkaz pro instalaci je:

```
# aptitude install openssh-server
```

Jelikož se server bude připojovat na vzdálená zařízení, musí být doinstalován i balík pro klientskou část *openssh-client*. Balík zajišťuje klientskou část pro připojení na vzdálená zařízení a umožňuje generování klíčového páru. Příkaz pro instalaci je:

```
# aptitude install openssh-client
```

7.2.1 Nastavení SSH serveru

Po instalaci je provedeno zabezpečení a konfigurace SSH serveru. Serverovou část zajišťuje daemon **sshd** a konfigurační soubor se nachází ve složce */etc/ssh/sshd_config*. V konfiguračním souboru se nachází mnoho předvoleb. Popisovaný jsou aktivní nastavení. Přehled nastavení SSH serveru na *pepina*:

```
#Port na kterém poslouchá SSH server
```

```
Port 22
```

```
#Využití protokolu verze 2
```

```
Protocol 2
```

```
#Místo kde jsou uloženy klíče HostKeys protokolu verze 2
```

```
HostKey /etc/ssh/ssh_host_rsa_key
```

```
HostKey /etc/ssh/ssh_host_dsa_key
```

```
#Přihlášení je provedeno jako oddělený proces
```

```
UsePrivilegeSeparation yes
```

```
#Životnost klíče (v sekundách) a velikost klíče
```

```
KeyRegenerationInterval 3600
```

```
ServerKeyBits 768
```

```
#Úroveň logování do syslog a do logu
```

```
SyslogFacility AUTH
```

```
LogLevel INFO
```

```
#Nepřihlási-li se uživatel, po uplynutí času (v sekundách) bude odpojen
```

```
LoginGraceTime 120
```

```
#Možnost přihlásit se jako root
```

PermitRootLogin yes

#Kontrola nastavení práv u uživatele
StrictModes yes

#Povolení přihlašování přes RSA
RSAAuthentication yes

#Povolení ověření pomocí klíčů
PubkeyAuthentication yes

#Nedovoluje číst data z .rhosts a .shost
IgnoreRhosts yes

#Nedovoluje připojení rhost přes RSA
RhostsRSAAuthentication no

#Nedovoluje připojení přes rhosts
HostbasedAuthentication no

#Připojení s prázdným heslem není povoleno
PermitEmptyPasswords no

#Není povolena výzva - odpověď pro heslo
ChallengeResponseAuthentication no

#Dovoluje tunelovaná pro X11
X11Forwarding yes

#Udává číslo displeje, které je k dispozici pro tunelování
X11DisplayOffset 10

#Nedovoluje vypsání uvítací výpis
PrintMotd no

#Vypisuje údaje o posledním přihlášení při přihlášení
PrintLastLog yes

#Zajišťuje informace o tom, že klient komunikuje se serverem

```
TCPKeepAlive yes
```

```
#Umožňuje klientovi předat lokální proměnné prostředí  
AcceptEnv LANG LC_*
```

```
#Povoluje Pluggable Authentication Module interface  
UsePAM yes
```

```
Subsystem sftp /usr/lib/openssh/sftp-server
```

7.2.2 Připojení na vzdálené zařízení pomocí hesla

Přihlášení za pomoci hesla odpovídá metodě D z teoretické části o SSH. Pro přihlášení je nutno znát uživatelské jméno, heslo a adresu vzdáleného zařízení. Případně je také potřeba zadat i číslo portu, pokud není využíván standardní port 22.

```
$ ssh -p cislo_portu uzivatel@server
```

Po připojení bude uživatel vyzván k zadání hesla, poté bude spojení navázáno.

V případě využití zabezpečeného kopírování ze vzdáleného zařízení pomocí **scp** je syntaxe příkazu následující.

```
$ scp uzivatel@server:/cesta_na_vzdalenem_zarizeni/nazev_souboru /cesta_ na_mistnim_zarizeni
```

Pokud se kopíruje soubor na vzdálené zařízení, je zadání příkazu:

```
$ scp /cesta_na_mistnim_zarizeni/nazev_souboru uzivatel@server:/cesta_ na_vzdalenem_zarizeni
```

Opět po zadání příkazu je uživatel vyzván k zadání hesla.

7.2.3 Generování privátního a veřejného klíče

Generování privátního a veřejného klíče probíhá pomocí programu **ssh-keygen**. Jednotlivé nastavení je možno zadávat přes parametry. Mezi nejdůležitější patří **-t**, který určuje typ šifry. Pro šifrování lze využít šifru RSA nebo DSA. Dalším neméně důležitým parametrem je **-b**. Pomocí něj je určena síla šifry. Při zadání příkazu dojde k vytvoření klíčového páru. Privátní klíč lze šifrovat pomocí vybraného hesla (*passphrase*). Na toto heslo bude klient vždy dotázán při přístupu na vzdálené zařízení. V případě, že se přes SSH bude přistupovat přes automatizovaný skript je vhodné *passphrase* nezadávat.

```

$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/thomas/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/thomas/.ssh/id_rsa.
Your public key has been saved in /home/thomas/.ssh/id_rsa.pub.
The key fingerprint is:
f5:dc:1f:2f:07:b3:39:1c:d5:da:af:6e:25:b0:b6:c4 thomas@idefix
The key's randomart image is:
+--[ RSA 2048]-----+
|           |
|           .|
|          . o|
|         . o.. +|
|        S  .oo*..|
|           E..B=|
|           o .*o=|
|           . .= |
|           oo  |
+-----+

```

Pokud již jsou klíče vygenerovány, je nutné nahrát veřejný klíč na vzdálené zařízení. Pro nakopírování lze využít příkaz **ssh-copy-id**. Přepínač **-i** určuje kde hledat klíč.

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub thomas@192.168.1.2
```

8 ÚKOLY SYSTÉMU

8.1 Zprovoznění rsyslogu

Zálohovací server má funkci shromaždiště systémových událostí (logů) ze vzdálených zařízení, které mají nastaveno odeslání těchto hlášení na zálohovací server. O chod logování v Debianu Squeeze se stará program **rsyslog** a jeho daemon **rsyslogd**. Získané logy se ukládají do složky */var/log/rsyslog*, kde jsou dále tříděny podle daných kritérií.

8.1.1 Nastavení rsyslogu

Konfigurace se provádí v souboru */etc/rsyslog.conf*, kde je nastavení **rsyslogu**. Popisováni jsou pouze změny konfigurace v konfiguračním souboru.

Daemon **rsyslogd** naslouchá na portu 514 pro spojení typu TCP a UDP. Příkaz **ModLoad** zajišťuje, jaký druh spojení bude použit. Pro TCP je to **imtcp** a pro UDP **imudp**. Naslouchání služby na určitém portu je možné příkazem pro TCP **InputTCPServerRun** a UDP **UDPServerRun**. Adresa UDP serveru je definována jako **UDPServerAddress 0.0.0.0**. Z toho vyplývá, že naslouchá všem příchozím spojením pro UDP na portu 514. Spojení jsou povolena jen pouze na IP adresy, které jsou zahrnuty jako povolené v **iptables**.

Při vytváření logu jsou nastavena uživatelská práva, aby se k nim mohl dostat jen povolený uživatel. Vlatníkem souboru je uživatel **root** patřící do skupiny **adm**. Přijaté hlášení jsou ukládány do složky */etc/rsyslog*, kde dále je vytvořena složka (*%HOST-NAME%*) s názvem vzdáleného zařízení. V této složce se pak vytváří příslušný log soubor ve tvaru ROK-MESIC-DEN.log.

```
#####  
#### MODULES ####  
#####  
  
# provides UDP syslog reception  
$ModLoad imudp  
$UDPServerAddress 0.0.0.0  
$UDPServerRun 514  
  
# provides TCP syslog reception  
$ModLoad imtcp  
$InputTCPServerRun 514  
  
#####  
#### GLOBAL DIRECTIVES ####  
#####  
$FileOwner root  
$FileGroup adm  
$FileCreateMode 0640  
$DirCreateMode 0755  
  
#####  
#### RULES ####  
#####  
$template DynaFile, "/var/log/rsyslog/%HOSTNAME%/%$YEAR%-%$MONTH%-%$DAY%.log"
```

```
-nms  
*. * -?DynaFile
```

8.2 Zálohování log souborů

Zálohování log souborů je prováděno naprogramovanými BASHovými skripty, které se starají o kopírování a následnou archivaci na zálohovacím serveru. Pravidelné spuštění skriptů zajišťuje **cron**, který se spouští ve stanovenou dobu.

Zálohování je prováděno ze zhruba 40 vzdálených zařízení jednou denně. Velikost jednotlivých log souborů je okolo 50MB pro každé vzdálené zařízení. Log soubory jsou pro svou velikost komprimovány. Soubory starší 180 dnů jsou mazány.

Před samotným procesem zálohování je nutné vygenerovat privátní a veřejný a následně je rozkopírovat na vzdálená zařízení. Z důvodu bezpečnosti je lépe mít pro každé vzdálené zařízení jiný klíč.

8.2.1 Zálohovací skript

Zálohovací skript slouží ke kopírování a archivaci log souborů ze vzdálených zařízení. Archivuje se ukládáním do složek ve tvaru *adresa_vzdaleneho_zarizeni/rok/mesic*. Každý den se po dávce zálohování odesílá email s průběhem zálohování. Funkce skriptu je rozdělena na dvě části. Tato funkce se rozlišuje pomocí parametrů. V případě špatného zadání parametru se vypíše nápověda pro vložení správných parametrů.

- **Zálohování vzdálených zařízení** - parametr -f
- **Zálohování mailového serveru** - parametr -m

Před prvním spuštěním skriptu je potřeba nastavit práva pro spuštění. Práva se přidávají příkazem:

```
$ chmod +x backup-logs.sh
```

Spuštění skriptu pro zálohování vzdáleného zařízení se spouští s jednotlivými parametry:

```
$ ./backup-logs.sh -f nazev_souboru /absolutni_cesta/vzdaleneho_zarizeni  
/absolutni_cesta/mistniho_zarizeni emailova@adresa.cz
```

Název skriptu je **backup-logs.sh**. Dále následuje přepínač **-f** označující zálohování vzdáleného zařízení. Další částí je **nazev_souboru**, který určuje název zálohovaného souboru na vzdáleném zařízení. Parametr **/absolutni_cesta/vzdaleneho_zarizeni**

je cesta k zálohovanému souboru na vzdáleném zařízení. Předposlední parametr **/absolutni_cesta/ mistniho_zarizeni** odkazuje na složku na zálohovacím serveru, kde bude zálohovaný soubor uložen. Z parametru **emailova@adresa.cz** se převezme email a výsledek zálohování (log soubor) se odešle na danou adresu. Parametry jsou přebírány a jsou dále využívány ve skriptu.

Před samotným zálohováním se vygeneruje seznam vzdálených zařízení. Skript volá pomocný skript **dns-tz.sh**, který pomocí příkazu **host -l domena.cz | grep 192.168** vyfiltruje seznam vzdálených zařízení z DNS a uloží ho do souboru *dns-list.txt*. Úspěšné či neúspěšné získání seznamu z DNS serveru je logováno do vlastního zálohovacího log souboru. Příkaz **sed** ze souboru *dns-list.txt* načte seznam a odstraní z něj řádky, které nerepresentují vzdálené zařízení pro zálohování. Výsledek úprav je uložen do nového souboru *dns-list.txt*. Pomocí **awk** se provede poslední úprava. Jelikož seznam obsahuje ještě nepotřebné sloupce s nedůležitými informacemi, musí se tyto sloupce odstranit. Příkaz **cat** načte soubor *dns-list.txt* a **awk** vybere ze souboru první sloupec, který obsahuje doménovou adresu vzdáleného zařízení a výsledek je uložen do nového souboru *servery.txt*.

Zálohování probíhá načítáním příslušného vzdáleného zařízení ze souboru *servery.txt*. Po načtení vzdáleného zařízení se kontroluje, zda existuje složka serveru, kde se budou zálohy ukládat. Zde se využívá parametru **/absolutni_cesta/mistniho_zarizeni**. Jak bylo zmíněno, archivace souboru se provádí ve tvaru *adresa_vzdaleneho_zarizeni/rok/mesic*. Skript kontroluje existenci složky *adresa_vzdaleneho_zarizeni*. V případě, že složka není vytvořena, tak ji skript vytvoří. Obdobný postup platí pro složky *rok* a *měsíc*.

Skript má vytvořeny složky, ve kterých bude zálohovaný soubor umístěn. Kopírování souboru na zálohovací server se využívá zabezpečeného kopírování **scp**. Název zálohovaného souboru (parametr **nazev_souboru**) a umístění na vzdáleném zařízení (parametr **/absolutni_cesta/vzdaleneho_zarizeni**) se bere z parametrů při spouštění skriptu. Uživatelé na vzdálených zařízeních mají stejné jméno. Přihlašování na vzdálený server probíhá pomocí ověřování klíčového páru (privátního a veřejného klíče). Informace o úspěšném nebo neúspěšném zkopírování se loguje do vlastního log souboru. Jelikož zálohovaný soubor má jednotné jméno pro všechny vzdálená zařízení, je potřeba ho přejmenovat na vhodný tvar. Po přejmenování má tvar *ROK-MESIC-DEN_domena_adresa.gz*. Příkaz **mv** umožňuje přejmenování souboru. Událost přejmenování se také loguje do vlastního log souboru.

Pokud je vše vykonáno, přechází skript na další řádek souboru v *servery.txt* a algoritmus zálohování se opakuje do té doby, dokud se nepřipojí na všechna určená zařízení.

Po celkovém zálohování se odešle email na emailovou adresu (parametr **emailova@adresa.cz**), ve kterém je obsah vlastního log souboru celého zálohování.

Spouštění skriptu pro zálohování mailového serveru se spouští s parametry:

```
$ ./backup-logs.sh -m nazev_souboru /absolutni_cesta/vzdaleneho_zarizeni
/absolutni_cesta/mistniho_zarizeni emailova@adresa.cz
```

U zálohování mailového serveru se negeneruje seznam vzdálených zařízení, ale pouze se ručně edituje soubor *mail-servery.txt*. Další změnou oproti předchozímu řešení je změna názvu zálohovaného souboru. Zálohovaný soubor nemá konstantní název, ale mění se v závislosti na datumu. Datum je vždy z předchozího dne, kdy se vytváří log soubor na mailovém serveru. Zálohovaný soubor je na mailovém serveru již připraven s daným názvem a proto se neprovádí zálohování. Proto je nutné použít příkaz **date -d '-1 day' + %d**, který zajistí odečtení jednoho dne ode dne, kdy se zálohuje.

Další postupy jsou obdobné jako u zálohování vzdálených zařízení.

8.2.2 Nastavení crontabu

Cron se stará o pravidelné spouštění zálohovacího skriptu a také má na starosti mazání souborů, které dosáhly stáří 180 dnů. Vložení záznamu do crontabu se provede příkazem:

```
$ crontab -e
```

Skript pro zálohování vzdálených zařízení se bude pravidelně spouštět každý den v týdnu v 11 hodin 0 minut. Záznam v **crontabu** bude následující:

```
0 11 * * * /home/thomas/backup-logs.sh -f contrackd-stats.log.1.gz
/var/log /home/thomas/skript emailova@adresa.cz
```

Zálohování logů z mailového serveru probíhá každý den v týdnu v 10 hodin a 0 minut. Vložený záznam bude:

```
0 10 * * * /home/thomas/backup-logs.sh -m -mail.log.bz2 /var/log/mail
/home/thomas/skript emailova@adresa.cz
```

Mazání starých logů ze vzdálených zařízení starších než 180 se provádí každý den v týdnu ve 12 hodin a 0 minut. **Cron** vyhledá ve složce */home/thomas/logs* všechny soubory, které vyhovují podmínce staří 180 dnů a jejich název obsahuje řetězec „*.domena.cz.gz“. Záznam pro mazání je:

```
0 12 * * * find /home/thomas/logs -type f -name
'*.domena.cz.gz' -mtime +180 -exec rm '{}' \;
```

Mazání starých logů z mailového serveru probíhá opět každý den v týdnu v 13 hodin a 0 minut. Vložený záznam a jeho princip je obdobný jako u vzdáleného serveru:

```
0 13 * * * find /home/thomas/logs -type f -name
'*-mail.log.bz2' -mtime +180 -exec rm '{} ' \;
```

Log soubory o zálohování (rozesílají se emailem) starší 180 dnů jsou mazány každé pondělí ve 14 hodin a 0 minut. Vložený záznam do **cronu** je:

```
0 14 * * 1 find /home/thomas/logs -type f -name
'zaloha_*.log' -mtime +180 -exec rm '{} ' \;
```

Pro kontrolu správnosti vložených údajů do **crontabu** je možné využít příkaz:

```
$ crontab -l
```

ZÁVĚR

V práci byla popisována tvorba kompletního vytvoření zálohovacího serveru od instalace linuxové distribuce, správy systému až po vytvoření samotného zálohovacího skriptu, který se staral o samotný proces zálohování. Cílem práce bylo vytvořit server, který slouží jako centralizované místo pro log soubory ze sítě internetového poskytovatele. Výhodou centralizace bylo snadné dohledávání časových souvislostí při správě. Instalace systému byla velmi snadná a intuitivní. V průběhu instalace nenastal závažný problém. Instalace byla přehledná a měla podrobný popis nabízených voleb. Vytváření pole RAID bylo nejtěžší částí instalace. Před samotnou instalací bylo nutné znát jak budou rozděleny jednotlivé diskové oddíly, a na kterých oddílech bude vytvořeno pole RAID. Samotné vytvoření pole bylo přehledné s přesnými popisy jednotlivých voleb. Po instalaci byl vytvořen logický oddíl LVM2, kde se ukládají zálohované log soubory. Zálohovací server byl zabezpečen pomocí pravidel **iptables**. Povolený přístup měly definované IP adresy a služby. Dalším bezpečnostním opatřením bylo to, že server neměl veřejnou IP adresu a byl pouze dosažitelný z vnitřní sítě poskytovatele. O pravidelné spouštění zálohovacího skriptu se staral **cron**, který také vykonával příkazy pro mazání starých souborů. Vzdálené připojení k serveru zajišťoval SSH a díky němu bylo možné server vzdáleně spravovat. Sběr hlášení a log souborů z bezdiskových vzdálených zařízení obsluhoval síťový **syslog** server a ukládal je na zálohovacím serveru. Na serveru běžely pouze ty služby, které byly nutné k chodu zálohovacího serveru a nebyly vykonávány žádné další úkoly nesouvisející se zálohováním.

Zálohovací skript byl napsán v příkazovém interpretu BASH a vykonává samotný akt zálohování. Staral se o kopírování na server, přejmenování log souboru na požadovaný tvar a uložení do složky. Skript obsahoval i pomocný skript pro získání seznamu vzdálených zařízení z DNS serveru pomocí zone transfer. Tato funkce byla dostupná pouze pro IP adresu zálohovacího serveru. Tímto bezpečnostním opatřením nehrozilo riziko zneužití DNS záznamů z vnější veřejné internetové sítě.

V průběhu psaní skriptu byla funkce získání seznamu vzdálených zařízení diskutována, zda ji využít nebo ne. Jelikož server nemá veřejnou IP adresu a je ve vnitřní síti, bylo rozhodnuto využít DNS jako zdroj seznamu zařízení. Síť poskytovatele se v budoucnu bude rozrůstat a počet vzdálených zařízení bude narůstat. Pomocí seznamu z DNS bylo zálohování plně automatizováno. Zálohovací skript byl rozdělen na dvě části. V první části se provádělo zálohování vzdálených zařízení a v druhé v části bylo prováděno zálohování mailového serveru. Jeho adresa se nezískávala z DNS transfer zone.

Volba OS GNU/Linux Debian se jevila jako správná. Oficiální dokumentace k jednotlivým částem distribuce je dostupná na webu a je přehledná a srozumitelná. Důležité aktualizace OS jsou vydávány často a chyby systému nejsou zamlčovány. V průběhu

testovacího provozu se zálohovací server jevil jako stabilní a nebylo nutné provést žádné potřebné dodatečné úpravy.

ZÁVĚR V ANGLIČTINĚ

The complete creation of a backup server (from installation of a Linux distribution and system administration to creation of a backup script, which provides backup process) is described in this work. The objective of this work was to create a server which serves as a centralized point in the network of an internet provider for storing of log files. The benefit from this centralization is that data can be easily found in context of time during the network administration.

System installation was smooth and intuitive. There wasn't any important problem during the installation. The RAID creation was only difficult part of the installation. It is necessary to know disk partitions and which the RAID will be created on. The RAID creation was transparent with precise description of available options. After installation the LVM2 logic partition was created. The log files are stored on this LVM2 logic partition.

The backup server was secured by the **iptables** rules. Only specified IP addresses and services had granted access. Another security step was that the server hadn't a public IP address and could be reached only from within provider's network. The **cron** regularly run backup script and performed commands for deletion of obsolete files. The remote access was provided by SSH and therefore it was able to administer the server remotely. The network **syslog** server collected messages and log files from remote devices without installed hard drive. Only services necessary for proper function of the backup server run on the server. No other tasks were performed.

Backup script was written in BASH shell and the script itself performs backup process. The backup script performed copying logs on the server, renaming log file according to naming convention and log storage to appropriate folder. The backup script contained an additional script for obtaining a list of remote devices from the DNS using zone transfer transaction. This function was accessible only for IP address of backup server. This safety precaution eliminated exploitation risk of DNS entries from outer Internet network.

During script creation we discussed deeply if this function for obtaining the list of remote devices will be part of the script or not. Because the backup server has not a public IP address and it is located on the internal network we decided to use the DNS entries as a source of a list of remote devices. The provider's network will expand in near future and the number of devices will increase too. Using DNS entries the backup process was fully automated. The backup script was divided into two parts. The first part performed backup process of remote devices and the second part performed backup process of a mail server. The IP address of this server was obtained from the DNS zone transfer transaction.

Choosing Debian GNU/Linux appeared to be proper. Official documentation for individual parts of OS distribution is available on the Web and it is transparent and understandable. Important OS updates are released quite often and OS bugs are not kept in silence. During the test operation the backup server appeared to be stable and it was not necessary to perform any supplemental modifications.

SEZNAM POUŽITÉ LITERATURY

- [1] PTÁČEK, Lubomír. *Linux : dokumentační projekt*. 4., aktualiz. vyd. Brno : Computer Press, 2007. 1334 s. ISBN 978-80-251-1525-1.
- [2] TURNBULL, James; LIEVERDINK, Peter; MATOTEK, Dennis. *Pro Linux System Administration*. 2009. Berkley : Apress, 2009. 1025 s. Dostupné z WWW: <http://books.google.com/books?id=BKn8mfYGfcAC&hl=cs&source=gbs_navlinks_s>. ISBN 978-1-4302-1913-2.
- [3] WELSH, Matt; KAUFMAN, Lar. *Používáme Linux*. 2. vyd. Praha : Computer Press, 1997. 612 s. ISBN 8072260014.
- [4] Linux. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 29. 12. 2002, last modified on 9. 2. 2011 [cit. 2011-02-10]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Linux>>.
- [5] *Root.cz* [online]. 07.11.2006, 07.11.2006, 05:59 [cit. 2011-05-25]. Main / Kernel. Dostupné z WWW: <<http://wiki.root.cz/Main/Kernel>>.
- [6] *Debian - Univerzální operační systém* [online]. 2.13. 2010, 8.10. 2010 [cit. 2011-03-14]. A Brief History of Debian - Introduction – What is the Debian Project? . Dostupné z WWW: <<http://www.debian.org/doc/manuals/project-history/ch-intro.en.html>>.
- [7] KRČMÁŘ, Petr. *Linuxová distribuce Debian*. In KRČMÁŘ, Petr. *Linuxová distribuce Debian* [online prezentace]. 2009 [cit. 2011-03-02]. Dostupné z WWW: <http://lvb.sti.fce.vutbr.cz/public/LinuxAlt_2009/2009_11_07_LA_05_Debian/2009_11_07_LA_05_Debian.pdf>.
- [8] *Debian - Univerzální operační systém* [online]. 11.3. 2011 [cit. 2011-03-14]. Debian logos. Dostupné z WWW: <<http://www.debian.org/logos/>>.
- [9] TRKOLA, Ota. *Debian-linux.cz : zpravodajství z distribuce Debian* [online]. 6.2. 2011 [cit. 2011-03-03]. Debian 6.0 Squeeze byl vydán (Debian-linux.cz). Dostupné z WWW: <<http://www.debian-linux.cz/debian-6-0-squeeze-byl-vydan/>>.
- [10] MATTHEW, Neil; STONES, Richard; KRÁSENSKÝ, David. *Linux : začínáme programovat*. Brno : Computer Press, 2008. 829 s. ISBN 978-80-251-1933-4.
- [11] MASTERS, Jon; BLUM, Richard. *Linux profesionálně : programování aplikací*. Vyd. 1. Brno : Zoner Press, 2008. 539 s. ISBN 978-80-86815-71-8.
- [12] NEMETH, Evi; HEIN, Trent R; SNYDER, Garth. *Linux : kompletní příručka administrátora*. Vyd. 1. Brno : Computer Press, 2004. 828 s. ISBN 8072269194.

- [13] *Debian Hypertext Man Pages* [online]. 22.9. 2010 [cit. 2011-03-23]. Tar. Dostupné z WWW: <<http://manpages.debian.net/cgi-bin/man.cgi?query=tar&apropos=0&sektion=0&manpath=Debian+6.0+squeeze&format=html&locale=en>>.
- [14] *Debian Hypertext Man Pages* [online]. 2002 [cit. 2011-03-23]. Gzip. Dostupné z WWW: <<http://manpages.debian.net/cgi-bin/man.cgi?query=gzip&apropos=0&sektion=0&manpath=Debian+6.0+squeeze&format=html&locale=en>>.
- [15] *Debian Hypertext Man Pages* [online]. 2011 [cit. 2011-03-23]. Bzip2. Dostupné z WWW: <<http://manpages.debian.net/cgi-bin/man.cgi?query=bzip2&apropos=0&sektion=0&manpath=Debian+6.0+squeeze&format=html&locale=en>>.
- [16] *Debian Wiki* [online]. 2010, 2010-05-03 [cit. 2011-03-28]. Ext3. Dostupné z WWW: <<http://wiki.debian.org/Ext3>>.
- [17] *Git.kernel.org* [online]. 2011, Mon Mar 28 07:45:27 2011 [cit. 2011-03-28]. Linux/kernel/git/stable/linux-2.6.29.y.git/blob. Dostupné z WWW: <<http://git.kernel.org/?p=linux/kernel/git/stable/linux-2.6.29.y.git;a=blob;f=Documentation/filesystems/ext3.txt>>.
- [18] *Debian Wiki* [online]. 2011, 2011-02-15 [cit. 2011-03-28]. Ext4. Dostupné z WWW: <<http://wiki.debian.org/Ext4>>.
- [19] *Git.kernel.org* [online]. 2011, Mon Mar 28 07:44:56 2011 [cit. 2011-03-28]. Linux/kernel/git/stable/linux-2.6.29.y.git/blob. Dostupné z WWW: <<http://git.kernel.org/?p=linux/kernel/git/stable/linux-2.6.29.y.git;a=blob;f=Documentation/filesystems/ext4.txt>>.
- [20] Red Hat Documentation [online]. 2010 [cit. 2011-03-28]. 5.6.2. RAID-Based Storage. Dostupné z WWW: <http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/4/html/Introduction_To_System_Administration/s2-storage-adv-raid.html#ftn.id3040124>.
- [21] DOČEKAL, Michal. *Linux E X P R E S* [online]. 3.12. 2009 [cit. 2011-03-28]. Jaké jsou typy polí RAID a jaké jsou jejich výhody a nevýhody. Dostupné z WWW: <<http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-raid-teoreticky>>. ISSN 1801-3996.
- [22] RAID. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 27.5. 2005, last modified on 26.3. 2011 [cit. 2011-03-30]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/RAID>>.

- [23] DOČEKAL, Michal. *Linux E X P R E S* [online]. 14.12. 2009 [cit. 2011-03-30]. Linux na serveru: LVM (Logical Volume Manager) a a diskové šifrování v podobě dm-crypt/LUKS. Dostupné z WWW: <<http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-lvm-a-diskove-sifrovani>>. ISSN 1801-3996.
- [24] BROŽ, Milan. *LVM - optimalizace a záchrana dat*. In BROŽ, Milan. *LVM - optimalizace a záchrana dat* [online]. [s.l.] : [s.n.], 2009 [cit. 2011-03-31]. Dostupné z WWW: <http://lvb.sti.fce.vutbr.cz/public/LinuxAlt_2009/2009_11_08_LA_04_LVM/2009_11_08_LA_04_LVM.pdf>.
- [25] Rackable Systems, Inc. *The Linux Documentation Project* : LVM HOWTO [online]. Revision 0.19. 2006, 2006-11-27 [cit. 2011-04-01]. What is LVM?. Dostupné z WWW: <<http://tldp.org/HOWTO/LVM-HOWTO/>>.
- [26] DOČEKAL, Michal. *Linux E X P R E S* [online]. 18.3. 2010 [cit. 2011-04-01]. Debian na Linuxovém serveru, LVM a snapshoty. Dostupné z WWW: <<http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-lvm-a-snapshoty>>. ISSN 1801-3996.
- [27] DOČEKAL, Michal. *Linux E X P R E S* [online]. 8.3. 2010 [cit. 2011-04-16]. Server s Debian Linuxem: Logical Volume Manager prakticky. Dostupné z WWW: <<http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-lvm-prakticky>>. ISSN 1801-3996.
- [28] KRČMÁŘ, Petr. *Linux : tipy a triky pro bezpečnost*. 1. vyd. Praha : Grada, 2004. 207 s. ISBN 8024708124.
- [29] KRČMÁŘ, Petr. *Linux : postavte si počítačovou síť*. 1. vyd. Praha : Grada, 2008. 182 s. ISBN 978-80-247-1290-1.
- [30] HONTANÓN, Ramón J; ROUBÍČEK, Ludvík. *Linux : praktická bezpečnost*. 1. vyd. Praha : Grada, 2003. 438 s. ISBN 8024706520.
- [31] DOČEKAL, Michal. *Linux E X P R E S* [online]. 10.5. 2010 [cit. 2011-04-04]. SSH na linuxovém serveru. Dostupné z WWW: <<http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-uvod-do-ssh-pro-spravce>>. ISSN 1801-3996.
- [32] VYSKOČIL, Michal. *Linux E X P R E S* [online]. 14.9. 2006 [cit. 2011-04-04]. SSH receptem na bezpečnost 2. Dostupné z WWW: <<http://www.linuxexpres.cz/praxe/ssh-receptem-na-bezpecnost-2>>. ISSN 1801-3996.

-
- [33] AOKI, Osamu. *Debian Reference* [online]. 2. verze. 2010-12-10 [cit. 2011-01-07]. Dostupné z WWW: <<http://www.debian.org/doc/manuals/debian-reference/>>.
- [34] RFC 1034. *Domain Concepts and Facilities*. [s.l.] : The Internet Engineering Task Force, November 1987. 55 s. Dostupné z WWW: <<http://www.ietf.org/rfc/rfc1034.txt>>.
- [35] RFC 1995. *Incremental Zone Transfer in DNS*. [s.l.] : The Internet Engineering Task Force, August 1996. 8 s. Dostupné z WWW: <<http://www.ietf.org/rfc/rfc1995.txt>>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

APT	Advanced Packaging Tool
BASH	Bourne Again Shell
BIOS	Basic Input-Output System
BSD	Berkeley Software Distribution
CIFS	Common Internet File System
deb	Debian
DNS	Domain Name System
FSF	Free Software Foundation
GNU	GNU's Not Unix (GNU Není Unix)
GPL	General public licence
gzip	GNU zip
HW	hardware
ICMP	Internet Control Message Protocol
IEFT	Internet Engineering Task Force
IP	Internet Protocol
LE	Logical Extend
LV	Logical Volume
LVM	Logical Volume Management
NAS	Network Attached Storage
NAT	Network Address Translation
NFS	Network File System
OS	operační systém
PE	Physical Extend
POSIX	Portable Operating System Interface
PV	Physical Volume
RAID	Redundant Array of Independent Disks
rpm	Red Hat Package Manager
smbfs	Server Message Block File System
SSD	Solid-state drive
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
vfat	V File Allocation Table
VG	Volume Group

SEZNAM OBRÁZKŮ

Obr. 1. logo	2
Obr. 2. logo	3
Obr. 3. <i>Maskot Linux tučnák Tux</i> [4]	13
Obr. 4. <i>Logo distribuce</i> [8]	14
Obr. 5. <i>Diskové pole RAID-0</i> [22]	23
Obr. 6. <i>Diskové pole RAID-1</i> [22]	23
Obr. 7. <i>Diskové pole RAID-4</i> [22]	24
Obr. 8. <i>Diskové pole RAID-5</i> [22]	24
Obr. 9. <i>LVM2 - hierarchie jednotlivých vrstev</i> [24]	26
Obr. 10. <i>Volba asistovaného dělení disku.</i>	43
Obr. 11. <i>Výběr oddílů do RAID pole.</i>	44
Obr. 12. <i>Rozdělení disků na oddíly a RAID pole.</i>	44
Obr. 13. <i>Instalace SSH serveru z instalátoru Debianu.</i>	48

SEZNAM TABULEK

Tab. 1. Parametry příkazu iptables [29]	30
Tab. 2. Příkazy editoru sed [1]	37
Tab. 3. Volby příkazu sed [1]	37

SEZNAM PŘÍLOH

- P I. Firewall skript pro iptables
- P II. Zálohovací skript - backup-logs.sh
- P III. Pomocný zálohovací skript (generování seznamu vzdálených zařízení z DNS)
- dns-tz.sh

PŘÍLOHA P I. FIREWALL SKRIPT PRO IPTABLES

```
#!/bin/sh

#####
# Firewall povoluje komunikaci jen s pocitaci, které mají povolenou IP.
# Komunikace je povolena pouze na povolené služby poskytované serverem.
#####

# hlavička potřebná pro spuštění pomocí /etc/init.d ve Squeeze
### BEGIN INIT INFO
# Provides:          firewall
# Required-Start:    $local_fs $network
# Required-Stop:     $local_fs $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: firewall
### END INIT INFO

# definování povolených IP a služeb
POVOLENE_IP="192.168.1.0/24 192.168.2.0/24"
POVOLENE_SLUZBY_TCP="ssh"
POVOLENE_SLUZBY_UDP="syslog"

# spuštění a vložení pravidel
start_firewall () {

    # definování základních pravidel (policy) pro tabulku filter
    iptables -P INPUT DROP
    iptables -P OUTPUT ACCEPT

    # definování pravidel
    # povolení pro spojení TCP
    for SLUZBY in $POVOLENE_SLUZBY_TCP
    do
        for IP in $POVOLENE_IP
        do
            iptables -A INPUT -p tcp -dport $SLUZBY -s $IP -j ACCEPT
        done
    done
done
```

```

# povoleni pro spojeni UDP
for SLUZBY in $POVOLENE_SLUZBY_UDP
do
    for IP in $POVOLENE_IP
    do
        iptables -A INPUT -p udp -dport $SLUZBY -s $IP -j ACCEPT
    done
done

# povoleni pristupu z localhosta vsechno
iptables -A INPUT -j ACCEPT -i lo

# povoleni ICMP ze vseh stroju
iptables -A INPUT -p icmp -j ACCEPT

# co neni povoleno na TCP, dostane RST, na UDP ICMP Port Unreachable
iptables -A INPUT -i eth0 -p tcp -j REJECT --reject-with tcp-reset
iptables -A INPUT -i eth0 -p udp -j REJECT \
        --reject-with icmp-port-unreachable
}

# pri zastaveni smaze puvodni pravidla a vlozi zakladni pravidla, vse povoleno
stop_firewall () {
    iptables -t filter -F
    iptables -t filter -P INPUT ACCEPT
    iptables -t filter -P OUTPUT ACCEPT
}

# rozhrani pri spousteni pres BASH
case "$1" in
start)
    echo "Spoustim firewall!"
    start_firewall
;;

```

```
stop)
    echo "Zastavuji firewall!!!"
    stop_firewall
    ;;

restart)
    echo "Restartji firewall!!!"
    stop_firewall
    start_firewall
    ;;

*)
    echo "Prikaz pro spusteni: firewall.sh start"
    echo "Prikaz pro zastaveni: firewall.sh stop"
    echo "Prikaz pro restart: firewall.sh restart"
    ;;

esac
```

PŘÍLOHA P II. ZÁLOHOVACÍ SKRIPT - BACKUP-LOGS.SH

```
#!/bin/bash

#####
#
# - skript zalohuje data ze vzdalenyh serveru
# - lze vybrat ze dvou typu zalohovani
# 1) zaloha logu vzdalenyh zarizeni, prepinač -f
# 2) zaloha logu mail serveru, prepinač -m
# - adresa se nacita ze souboru radek po radku
# - pro vzdalena zarizeni se generuje seznam zarizeni pomoci
# DNS zone transfer, zajistuje skript dns-tz.sh
# - pro mail servery se edituje soubor rucne
# - po prekopirovani vseh logu odesle mail s logem o prubehu kopirovani
# - v souboru s adresami musi byt na konci vlozen prazdny radek
#
# - spusteni skriptu a jednotlivy parametry
# backup-logs.sh -f nalez_souboru /absolutni_cesta/vzdaleneho_zarizeni
# /absolutni_cesta/mistniho_zarizeni emailova@adresa.cz
#
#####

# servery.txt - cisty soubor z adresou jednotlivy vzdalenyh zarizeni
# mail-servery.txt - cisty soubor z adresou jednotlivy mail serveru

# GLOBALNI PROMENNE
# nastaveni datumu do parametru
    DATUM=$(date +%F)
    ROK=$(date +%Y)
    MESIC=$(date +%m)
    DEN=$(date +%d)
    CAS=$(date +%T)

# nastaveni uzivatelskeho jmena
UZIVATEL="thomas"
```

```

# premapovani hodnot z paramametru do promennych
SOUBOR=$2 # nazev zalohovaneho souboru
ZALOHA=$3 # absolutni cesta na vzdalenem zarizeni
        # vcetne zalohovaneho souboru (na konci cesty bez /)
CESTA=$4 # absolutni cesta pro ulozeni na lokalnim disku
        # (na konci cesty bez /)
EMAIL=$5 # emailova adresa pro odeslani logu

# vetveni programu podle zadaneho prepince
# -f (zaloha vzdalene zarizeni)
# -m (zaloha mail server)

case "$1" in
    -f) # zaloha logu vzdalenyh zarizeni

# zavolani dns-tz.sh pro vytvoreni aktualniho seznamu vzdalenyh zarizeni
./dns-tz.sh

# nacteni souboru, cte radek po radku
cat servery.txt | while read PRIHLASENI;

do
    # vytvoreni slozky na zalohovacim serveru - pokud jiz neexistuje
    if [ -d $CESTA ]; then # adresar jiz existuje
echo "Slozka existuje! $CESTA"
    else
mkdir $CESTA;
echo "Uspesne vytvoreni slozky! $CESTA"
    fi

    # vytvoreni slozky stroje z ktereho se kopiruje
    if [ -d $CESTA/$PRIHLASENI ]; then # adresar jiz existuje
echo "Slozka existuje! $CESTA/$PRIHLASENI"
    else
mkdir $CESTA/$PRIHLASENI;
echo "Uspesne vytvoreni slozky! $CESTA/$PRIHLASENI"
    fi

```

```

# vytvoreni slozky s aktualnim datumem
if [ -d $CESTA/$PRIHLASENI/$ROK ]; then
echo "Slozka existuje! $CESTA/$PRIHLASENI/$ROK"
else
mkdir $CESTA/$PRIHLASENI/$ROK
echo "Uspesne vytvoreni slozky! $CESTA/$PRIHLASENI/$ROK"
fi

if [ -d $CESTA/$PRIHLASENI/$ROK/$MESIC ]; then
echo "Slozka existuje! $CESTA/$PRIHLASENI/$ROK/$MESIC"
else
mkdir $CESTA/$PRIHLASENI/$ROK/$MESIC
echo "Uspesne vytvoreni slozky! $CESTA/$PRIHLASENI/$ROK/$MESIC"
fi

# zalohovani na server s vypisem statistiky do log souboru
scp $UZIVATEL@$PRIHLASENI:$ZALOHA/$SOUBOR $CESTA/$PRIHLASENI/$ROK/
$MESIC/$SOUBOR 2>> $CESTA/zaloha_vz_$DATUM.log
# kopirovani, vlozi chybovou hlasku pri chybnem kopirovani

if [ `echo $?` == 0 ]; then # podle navratove hodnoty se rozhoduje,
# zda bylo uspesne zkopirovano
echo "$ROK-$MESIC-$DEN $CAS $PRIHLASENI: Uspesne zkopirovano!" >>
$CESTA/zaloha_vz_$DATUM.log
else
echo "$ROK-$MESIC-$DEN $CAS $PRIHLASENI: NEUSPESNE ZKOPIROVANO!!" >>
$CESTA/zaloha_vz_$DATUM.log
fi

# prejmenovani soboru na pozadovany tvar
mv $CESTA/$PRIHLASENI/$ROK/$MESIC/$SOUBOR $CESTA/$PRIHLASENI/$ROK/
$MESIC/$DATUM-$PRIHLASENI.gz >> $CESTA/zaloha_vz_$DATUM.log

if [ `echo $?` == 0 ]; then # podle navratove hodnoty se rozhoduje,
# zda bylo uspesne prejmenovani
echo "$ROK-$MESIC-$DEN $CAS $PRIHLASENI: Uspesne prejmenovani!" >>
$CESTA/zaloha_vz_$DATUM.log
else

```

```

echo "$ROK-$MESIC-$DEN $CAS $PRIHLASENI: NEUSPESNE PREJMENOVANI!!" >>
    $CESTA/zaloha_vz_$DATUM.log
fi

done

# odeslani souboru zalohaciho logu na mail
PREDMET="Zalohovaci log - vzdalene zarizeni `date +%F`"
mailx -s "$PREDMET" $EMAIL < $CESTA/zaloha_vz_`date +%F`.log

;;

-m) # zaloha mailovych log souboru

# osetreni datumu, jelikoz kopirovany log ma datum predchoziho dne
DEN_ZALOHY=`date -d '-1 day' +%d`

cat mail-server.txt | while read PRIHLASENI;

do
    # vytvoreni slozky na zalohovacim serveru - pokud jiz neexistuje
    if [ -d $CESTA ]; then # adresar jiz existuje
echo "Slozka existuje! $CESTA"
    else
mkdir $CESTA;
echo "Uspesne vytvoreni slozky! $CESTA"
    fi

    # vytvoreni slozky stroje z ktereho se kopiruje
    if [ -d $CESTA/$PRIHLASENI ]; then # adresar jiz existuje
echo "Slozka existuje! $CESTA/$PRIHLASENI"
    else
mkdir $CESTA/$PRIHLASENI;
echo "Uspesne vytvoreni slozky! $CESTA/$PRIHLASENI"
    fi

```

```

# vytvoreni slozky s aktualnim datumem
if [ -d $CESTA/$PRIHLASENI/$ROK ]; then
echo "Slozka existuje! $CESTA/$PRIHLASENI/$ROK"
else
mkdir $CESTA/$PRIHLASENI/$ROK
echo "Uspesne vytvoreni slozky! $CESTA/$PRIHLASENI/$ROK"
fi

if [ -d $CESTA/$PRIHLASENI/$ROK/$MESIC ]; then
echo "Slozka existuje! $CESTA/$PRIHLASENI/$ROK/$MESIC"
else
mkdir $CESTA/$PRIHLASENI/$ROK/$MESIC
echo "Uspesne vytvoreni slozky! $CESTA/$PRIHLASENI/$ROK/$MESIC"
fi

# zalohovani na server s vypisem statistiky do log souboru
scp $UZIVATEL@$PRIHLASENI:$ZALOHA/$ROK/$MESIC/$DEN_ZALOHY$$SOUBOR $CESTA/$PRIHLA
$CESTA/zaloha_mail_$DATUM.log # kopirovani, vlozi chybovou hlasku
# pri chybnem kopirovani

if [ `echo $?` == 0 ]; then # podle navratove hodnoty se rozhoduje,
# zda bylo uspesne zkopirovano
echo "$ROK-$MESIC-$DEN $CAS $PRIHLASENI: Uspesne zkopirovano!" >>
$CESTA/zaloha_mail_$DATUM.log
else
echo "$ROK-$MESIC-$DEN $CAS $PRIHLASENI: NEUSPESNE ZKOPIROVANO!!" >>
$CESTA/zaloha_mail_$DATUM.log
fi

done

# odeslani souboru zalohaciho logu na mail
PREDMET="Zalohovaci log - mail servery `date +%F`"
mailx -s "$PREDMET" $EMAIL < $CESTA/zaloha_mail_`date +%F`.log

;;

```

```
*) # pri spatnych parametrech zobrazeni napovedy
echo "SPATNE PARAMETRY!!"
echo "-----"
echo "Pro zalohu VZDALENEHO ZARIZENI je prepinač -f"
echo "Pro zalohu MAIL SERVERU je prepinač -m"
echo "Spravne zadani parametru:"
echo "backup-logs.sh -f nazev_souboru /absolutni_cesta/vzdaleneho_zarizeni /absolu
    mistniho_zarizeni emailova@adresa.cz"
;;
esac
```

PŘÍLOHA P III. POMOCNÝ ZÁLOHOVACÍ SKRIPT (GENEROVÁNÍ SEZNAMU VZDÁLENÝCH ZAŘÍZENÍ Z DNS) - DNS-TZ.SH

```
#!/bin/bash

#####
# - získání seznamu vzdálených zařízení z DNS
# za pomoci DNS transfer zone
# - po získání seznamu probíhá vypreparování potřebných
# údajů pro další zpracování zálohovacího skriptu
#####

# dns-list.txt surový soubor z DNS
# dns-edit.txt upravený surový soubor z DNS (odstranění nepotřebných adres)
# servery.txt čistý soubor z adresou jednotlivých vzdálených zařízení

# nastavení datumu do parametru
    DATUM=$(date +%F)
    ROK=$(date +%Y)
    MESIC=$(date +%m)
    DEN=$(date +%d)
    CAS=$(date +%T)

# zavolání dotazu na DNS pomocí DNS transfer zone
host -l domena.cz | grep 192.168 > dns-list.txt

if [ `echo $?` == 0 ]; then
    echo "$ROK-$MESIC-$DEN $CAS: Úspěšně získán seznam zařízení z DNS!" >>
        ~/$CESTA/zaloha_$DATUM.log
    echo "" >> ~/$CESTA/zaloha_$DATUM.log # vložení prázdného řádku
else
    echo "$ROK-$MESIC-$DEN $CAS: NEÚSPĚŠNĚ ZÍSKÁN SEZNAM ZAŘÍZENÍ Z DNS!!" >>
        ~/$CESTA/zaloha_$DATUM.log
    echo "" >> ~/$CESTA/zaloha_$DATUM.log
fi
```

```
# odstraneni nepotrebných adres z vypisu
sed '/domena.cz has address 192.168.1.49/d; /old-mesto.domena.cz has
    address 192.168.1.109/d; /test.domena.cz has address 192.168.1.35/d;
    ' dns-list.txt > dns-edit.txt
echo "" >> dns-edit.txt # vlozeni prazdneho radku na konec souboru

# vytvoreni seznamu vzdalenyh zarizeni pro zpracovani skriptu
cat dns-edit.txt | awk '{print $1}' > servery.txt

if [ `echo $?` == 0 ]; then
    echo "$ROK-$MESIC-$DEN $CAS: Uspesna filtrace seznamu zarizeni!" >>
        ~/$CESTA/zaloha_$DATUM.log
    echo "" >> ~/$CESTA/zaloha_$DATUM.log # vlozeni prazdneho radku

else
    echo "$ROK-$MESIC-$DEN $CAS: NEUSPESNA FITLRACE SEZNAMU ZARIZENI!!" >>
        ~/$CESTA/zaloha_$DATUM.log
    echo "" >> ~/$CESTA/zaloha_$DATUM.log
fi
```