Souborové systémy OS Linux

Obsah

1 Úvod

Vývoj operačních systémů a externích datových médií s náhodným přístupem si vynutil důmyslnou organisaci dat. Motivací pro organisaci dat na jejich nosiči je několik. Jednou z nejstarších motivací, kterou dnes chápeme jako samozřejmou, je možnost *diferencovat data* do souborů — relativně samostatných celků. Budeme-li chápat tuto motivaci jako původní, bezprostředně potom se objevuje podobná. Je to snaha naopak shlukovat soubory, podle jejich logického obsahu, případně chronologického kontextu. Takovým aparátem jsou složky a adresáře. Pojem adresář neměl vždy v historii operačních systémů takovou podobu, na jakou jsme zvyklí nyní, ale téměř vždy sloužil právě jako „kontainer“ shlukující soubory, které k sobě nějakým způsobem patří.

Výše popsané motivace vedly k vytváření **souborových systémů**. S nástupem výkonných operačních systémů, které začaly být *multiuživatelské* a *multiúlohové*, se začaly vršit další požadavky i na souborové systémy. V dnešní době musí souborové systémy splňovat mnohá kritéria. Jednak je to potřeba dokonalé organisace místa na disku, která šetří prostorem, jednak je to například potřeba dostatečného výkonu u kritických aplikací. Zanedbatelná není ani škálovatelnost a možnost transakčního zpracování dat a metadat, se kterými souborový systém pracuje. Kromě klasických souborových systémů, se začínají objevovat i významné distribuované souborové systémy, umožňující sdílení dat buďto na logické nebo na fyzické úrovni.

Kromě robustních souborových systémů, jako je například XFS, původně vyvinutý v laboratořích fy SGI jako souborový systém pro superskalární počítače, existují i miniaturní souborové systémy, které je zejména vhodné umísťovat do malých a přenosných zařízení jako jsou například Flash EEPROM karty nebo mobilní telefony a tak dále.

Jelikož má operační systém Linux své zastoupení od výkonných serverů až po kapesní počítače, nutně si nemůže vystačit s jedním souborovým systémem. Kromě aspektů nasazení existují i historická hlediska. V dobách, kdy operační systém Linux vznikal nikdo netušil, jakého masivního nasazení se mu dostane, proto nyní nejrozšířenější a nejpoužívanější souborový systém EXT2 trpí řadou nedostatků, které byli při jeho vývoji a jeho plánovaném nasazení nepodstatné. Typickým příkladem je třeba journalování metadat. Při výpadku proudu na serveru s 200 GB diskovým polem trvá jeho restaurování něco kolem deseti hodin. Například při použití journalového systému ReiserFS se tato doba zkrátí na několik minut. Při implementaci EXT2 ale nikdo nepředpokládal, že někdy Linux poběží na větším disku než 1GB, proto neměla implementace journálu valný význam.

2 Manažer logických svazků

Logický manažer svazků (*Logical Volume Manager*, dále jen *LVM*) je mechanismus, který přidává další vrstvu mezi diskovou oblast a flexibilním způsobem umožňuje manipulovat se souborovými systémy. Celý projekt byl inspirován manažerem svazků ze systému HP UNIX, proto je téměř shodné i jeho softwarové rozhraní. Projekt LVM sestává z kódu v jádře Linuxu, dodatečných obslužných programů a dokumentace. LVM je implementováno pro jádra řady 2.4.x a lze jej najít na stránce http://linux.msede.com/lvm/.

Cílem projektu LVM bylo implementovat flexibilní subsystém *virtuálních disků* umožňující spravovat diskový prostor na okamžitě alokovat, případně zmenšovat prostor na přidělených discích. To vše bez nutnosti restartu systému, nebo jiné destruktivní činnosti.

Historicky vzato je velikost diskové oblasti (*Disk Partition*) statická. Tato vlastnost vyžaduje velkou obezřetnost při instalace software. Instalátor systému na sebe bere odpovědnost za navržení vhodné velikosti diskových oblastí. V potaz nemohou být brány pouze aktuální požadavky, nýbrž i požadavky do budoucna. Otázky kladené při instalaci by tedy neměly být: „Kolik dat budu na této oblasti mít?“, nýbrž spíš: „Kolik kdy budu potřebovat místa na této oblasti?“. Pokud systém zaplní diskovou oblast, je obvykle nutné destruktivně předělat diskové oblasti na celém médiu, nebo si pomoci jiným mechanismem, například zkopírovat část dat jinam a zavést symbolické odkazy.

Operační systémy vyšších tříd poměrně brzy opustily statické diskové oblasti. Většina operačních systémů vycházejících z UNIXu má schopnost rozčlenit fyzické disky na jistý počet jednotek. Takové jednotky, obecně z různých disků, mohou být potom shlukovány do „logických svazků“, v nich mohou být dále alokovány diskové oblasti. Jednotlivé jednotky mohou být přidávány a odebírány z logických svazků a tím je možné zvětšovat a zmenšovat jednotlivé diskové oblasti. To je, zjednodušeně řečeno, princip LVM.

2.1 Proč LVM?

V dnešní době není neobvyklé, že v Linuxovém serveru najdeme 3 až 10, nebo dokonce 50 až 60 disků. Například centrum genetického výzkumu v Izraeli disponuje několika servery s 90-ti diskovým polem. Správa takového množství disků a diskových oblastí není jednoduchá. Navíc se obvykle požaduje, aby bylo možné za běhu přidat disky a nafouknout jednotlivé svazky o jejich prostor.

Díky LVM je vytvořena dodatečná vrstva mezi fyzickou periferií a I/O rozhraním v jádře. Pomocí této vrstvy je možné pohlížet na celkový prostor z logického hlediska. LVM v zásadě umožňuje zřetězovat jednotlivé disky z diskového pole a pohlíží na ně jako na nová logická zařízení. Výhody jsou evidentní, od zjednodušení správy až po celkovou vyšší pružnost. Ztráty výkonu přidáním další vrstvy jsou minimální.

2.2 Základní pojmy

Pro pochopení dalšího textu si nejprve vysvětlíme některé základní pojmy.

- **Fyzické medium (physical media)**

  V tomto kontextu chápeme fyzické médium jako něco, na co si „lze sáhnout“. Každé fyzické médium je representováno svým speciálním zařízením v adresáři `/dev`. Fyzickými médii mohou být například diskové oblasti `/dev/hda1`, `/dev/hda2`, `/dev/sda1`, `/dev/sda1`, , ale například i speciální zařízení pro softwarový RAID, tedy `/dev/md0`, `/dev/md1` a tak dále.

- **Fyzický svazek (physical volume)**

  Fyzický svazek je *fyzické médium*, které má v sobě navíc zapsaná nutná administrativní data. Fyzický svazek je základním stavebním kamenem LVM.

- **Fyzický extent (physical extent)**

  Fyzický extent je alokační jednotka, lze si jej představit jako „velký blok“. Typicky je velikost od 4 do 256MB. Fyzický extent je umístěn na *fyzickém svazku*.

- **Skupina svazků (volume group)**

  V tomto místě už český překlad poněkud pokulhává, ale dle mého názoru je to asi jeden z nejpřijatelnějších překladů. Jednotlivé *fyzické extenty* mohou být přiřazeny do *svazku extentů*. Při definici takového svazku lze definovat i požadovanou velikost extentu a samozřejmě fyzické svazky, které budou skupinu svazků tvořit.

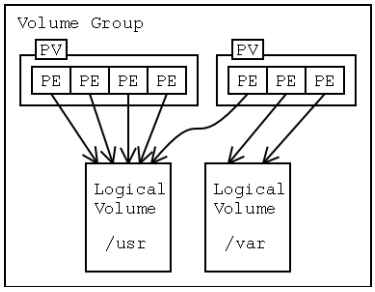
- **Logický svazek (logical volume)**

  V rámci jednotlivých *svazků extentů* mohou být jednotlivé extenty přiřazeny do *logických svazků*. Logický svazek lze v tomto kontextu chápat jako klasickou diskovou oblast, na logickém svazku lze přímo vytvářet souborové systémy nebo na něj třeba natvrdo uložit Oraclovskou databázi.

- **Souborový systém (file system)**

  Souborový systém je organisace souborů a adresářů, která je na logickém svazku.

Předešlá struktura, jakkoliv se může zdát komplikovaná, zaručuje velkou flexibilitu LVM. Před dalším výkladem si jednotlivé pojmy demonstrujme obrázkem.



Celý obrázek představuje jednu skupinu svazků, která sestává ze dvou fyzických svazků. Dva logické svazky, nesoucí souborové systémy pro `/usr` a `/var`, jsou alokovány z fyzických extentů, přitom logický svazek `/usr` obsahuje extenty z obou fyzických svazků.

2.3 Použití LVM

Nejprve je nutné mít zkompilovanou podporu LVM v jádru systému nebo mít ji k disposici jako modul. Ale vzhledem k tomu, že LVM je aktivní po celou bodu chodu systému, je mnohem lepší, mít jej přímo v jádře. V jádrem řady 2.4.x je LVM zahrnuto. Kromě jádra jsou potřeba ještě *user space utility*. Před připojením a aktivací LVM je totiž nutné detekovat všechny dostupné fyzické svazky a na samotnou manipulaci s LVM je třeba mít další obslužné programy.

V tuto chvíli se přímo nabízí kousavá otázka: *„Může být umístěn kořenový souborový systém na LVM?“*. Odpověď zní ano, ale není to vůbec jednoduché. Mechanismus jádra Linuxu umožňuje používat *initrd* filesystém. To je malý filesystém, který je zaveden do paměti, ještě před tím, než zavaděč, například `LILO`, předá řízení jádru. Jádro provede na takovém filesystému *initrd skript* a po jeho dokončení si remountuje nový kořenový filesystém. Pokud bude *initrd skript* obsahovat aktivaci LVM, pak jsme za vodou.

Ale obecně se doporučuje mít malý kořenový souborový systém, něco kolem 100MB úplně mimo LVM. Stejně tak odkládací oblast by měla být uložena mimo LVM. Jádro Linux si samo dokáže zřetězovat swapovací oblasti, případně na nich provádět *stripping*, pokud je to nutné a pokud mají odkládací oblasti nastavenu stejnou prioritu.

Pokud chceme použít LVM, musíme nejprve z patřičných fyzických médií vytvořit *fyzické svazky*. Předpokládejme, že máme disky, na kterém jsme vyhradili diskové oblasti. V ideálním případě by měl disk obsahovat pouze jednu diskovou oblast, nic ale nebrání tomu, aby na disku byly i jiné oblasti, třeba obsahující jiné operační systémy. Nejprve je nutné na diskové oblasti nastavit její typ. Například *linux native partition* je identifikována číslem 0x83, *linux swap partition* číslem 0x82. LVM je identifikována číslem 0x8e. Naneštěstí většina programů

typu `fdisk` ještě v sobě nemá tento údaj zanesen, takže je možné že po vložení hodnoty 0x8e bude nést oblast označení „unknown“.

Takže například pomocí programu `fdisk` se změní typ oblasti.

```
# fdisk /dev/hda
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 8e
Command (m for help): w
```

Při prvním použití LVM bychom měli spustit program `vgscan`, který LVM inicialisuje. V této fázi program nenajde žádná skupina svazků.

```
# vgscan
vgscan - reading all physical volumes (this may take a while...)
vgscan - no volume groups found
```

Nyní lze pomocí příkazu `pvcreate` inicialisovat fyzický svazek.

```
# pvcreate /dev/hda1
pvcreate - reinitializing physical volume
pvcreate - physical volume "/dev/hda1" successfully created
```

V tomto okamžiku je vytvořen jeden fyzický svazek — `/dev/hda1`. Fyzických svazků lze v systému tímto způsobem vytvořit samozřejmě víc. Fyzické svazky mohou být umístěny na stejném disku, i když zde by trochu unikal smysl, nebo na různých discích či diskových polích. Typické použití je například na RAID5 poli, speciální soubor `/dev/rd/c0d0`, represenující logický svazek RAID pole bude pro LVM jedním fyzickým svazkem.

Po vytvoření fyzických svazků lze přistoupit k vytvoření svazku extentů. Zjednodušeně řečeno, několik fyzických svazků se shlukuje a vytvoří se tak jedna skupina, obsahující několik fyzických svazků. Intuitivně si to lze například představit tak, že na dvou discích jsme vytvořili dva fyzické svazky a z nich jsme vytvořili jeden *volume group*. Volume group musí mít vždy své jméno. Například.

```
# vgcreate volg01 /dev/hda2 /dev/hdc1
vgcreate - INFO:using default physical extent size 4 MB
vgcreate - INFO: maximum logical volume size is 255.99 Gigabyte
vgcreate - doing automatic backupof volume group "volg01"
vgcreate - volume group "volg01" successfully created and activated
```

Předchozím příkazem jsme vytvořili *volume group* pojmenovanou `volg01`, která obsahuje extenty ze dvou fyzických svazků, velikost extentu lze ovlivnit jedním z argumentů programu `vgcreate`, implicitní velikost je 4MB. Stejně tak lze omezit i maximální velikosti a počet logických svazků. Toto omezení není bezúčelné, jeho vhodným nastavením lze snížit systémovou režii.

Během vytvoření svazku extentů je vytvořen i podadresář `/dev/volg01`. Ten bude do budoucna obsahovat další speciální soubory, které budou odpovídat zařízením jednotlivých logických svazků. Speciální soubor `/dev/volg01/group` umožňuje přistupovat k svazku extentů jako k blokovému zařízení. Stejně tak je aktualisována informace v souboru `/etc/lvmtab`, ve kterém jsou uchovány názvy všech svazků extentů, které byly systémem nadetekovány. Dále jsou vytvořeny dva binární konfigurační soubory, `/etc/lvmtab.d/volg01` a `/etc/lvmconf/volg01.conf`. Tyto soubory tvoří databázi pro ostatní příkazy LVM a je možné je znovu obnovit právě pomocí příkazu `vgscan`.

Jednotlivé atributy svazku extentů lze za běhy systému měnit pomocí příkazu `vgchange`, některé změny lze provést nedestruktivně, jiné destruktivně. Při přidání dalšího fyzického svazku do systému, třeba po přidání nového disku, je možné rozšířit skupinu svazků, k tomu slouží příkaz `vgextend`. Jednotlivé svazky extentů lze rovněž rozdělovat, slévat a podobně. Podobná užití si ukážeme dále.

Užitečná utilita je `vgdisplay`, umožňuje vypsat informace o svazku extentů. Může být použita ale až potom, co jsme aktivovali LVM pomocí programu `vgscan`. Příklad použití `vgdisplay`.

```
# vgdisplay -v volg01
--- Volume group ---
VG Name               volg01
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                4
Open LV               4
MAX LV Size           255.99 GB
Max PV                256
Cur PV                1
Act PV                1
VG Size               42.31 GB
PE Size               4 MB
Total PE              10832
Alloc PE / Size       0 / 0
Free  PE / Size       10832 / 42.31 GB
VG UUID               Wwzfu0-EJdJ-ENpy-KDsm-q4ba-4T6q-dFjnxl

--- Physical volumes ---
PV Name (#)           /dev/hda4 (1)
PV Status             available / allocatable
Total PE / Free PE    10832 / 10832
```

Údaje z výpisu informují o volném místu, o počtu extentů, jejich velikosti, atributech svazku a informují o počtu fyzických svazků, z nichž skupina sestává. Na fyzickém svazku, který lze intuitivně přirovnat k „disku se spoustou šikovných schopností“, lze vytvářet *logické svazky* — což jsou vlastně „diskové oblasti se spoustou šikovných vlastností“. Pro manipulaci s logickými svazky existují příkazy prefixované `lv`, pomocí `lvcreate` lze vytvořit nový logický svazek. Viz příklad.

```
# lvcreate -L 800M -n oracle_bin volg01
lvcreate - doing automatic backup of "volg01"
lvcreate - logical volume "/dev/volg01/oracle_bin" successfully created

# lvcreate -L 70G -n oracle_data volg01
lvcreate - doing automatic backup of "volg01"
lvcreate - logical volume "/dev/volg01/oracle_data" successfully created
```

Předchozími příákazy jsme vytvořili dva logické svazky, jeden o velikosti 800MB, druhý o velikosti 70GB. Každý logický svazek musí být pojmenován, jeho jméno musí být v rámci jedné skupiny svazků unikátní. S každým logickým svazkem LVM vytvoří i speciální soubor v adresáři `/dev/volg01`, viz předchozí příklad. Takový speciální soubor je speciální soubor blokového zařízení, které již mohou používat další *user space* programy, typicky třeba program `mount`.

Po vytvoření potřebných logických svazků je dobré do inicialisačních skriptů systému přidat volání programu `vgscan`. Rovněž se dobré při inicializaci změnit status všech skupin svazků pomocí volání `vgchange -a y`, tím oznámíme jádru, že skupiny jsou k disposici. Naopak, při zastavení systému je vhodné svazky odhlásit od jádra pomocí `vgchange -a n`.

Na speciálních zařízeních logických svazků je možné vytvářet souborová systémy a je možné mountovat je a pracovat s nimi. Například na servery **phoenix** je použito LVM v kombinaci s journalovým souborovým systémem ReiserFS. Harddisk serveru je rozdělen na následující oblasti.

```
   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1        16      8032+  83  Linux
/dev/hda2            17      1008    499968   83  Linux Swap
```

```
/dev/hda3          1009      1318    156240   83  Linux
/dev/hda4          1319     89355  44370648   8e  Unknown
```

První disková oblast obsahuje jádro systému. Jádro nemůže být uloženo na ReiserFS, protože tento souborový systém nezaručuje pevnou posici na souboru na disku. Další disková oblast je swapovací, následuje 160MB velká oblast pro kořenový souborový systém. Zbytek disku, cca 40GB je fyzický svazek LVM. V systému byl vytvořena jedna skupina svazků — `volg01`, která zatím obsahuje jen jediný fyzický svazek `/dev/hda4`. Ve skupině `volg01` byly vytvořeny logické svazky, které jsou namountovány na jednotlivé části souborového systému Linuxu.

```
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda3               156212     64436     91776  41% /
/dev/volg01/usr        6291260   1014856   5276404  16% /usr
/dev/volg01/var       10485436     67604  10417832   1% /var
/dev/volg01/home      26541264    313124  26228140   1% /home
alpha:/home/student    2055240   1664152    391088  81% /home/student
alpha:/home/staff      4110480   1056192    628352  63% /home/staff
```

2.4 Zvětšení svazků

Jednou z hlavních motivací, proč LVM je možnost co možná nejjednodušším způsobem měnit velikost svazku. V reálném provozu se občas stává, že některý svazek se zaplní, náchylné jsou zejména například *spool svazky*, kde se hromadí pošta. Taková Anna_Kurnikova.vbs, ta vám zaneřádí disk, ani nevíte jak. S požadavkem zvětšitelnosti je nutný i požadavek zmenšitelnosti média, svazek se v krajním případě může zvětšit na úkor jiného svazku.

Uvažujeme-li architekturu LVM tak, jak byla nastíněna, je jasné, že manipulace s velikostí lze provádět na úrovni logických svazků, ale i na úrovni skupin svazků, tedy o jednu úroveň níž. Je evidentní, že pokud přidáme do skupiny svazků další fyzický svazek, nepovede to samo o sobě ke zvětšení žádného logického svazku. Ve skupině svazků se pouze objeví nějaké nové volné extenty.

Na úrovní skupin svazků lze zvětšovat a redukovat skupinu o další fyzické svazky. Pomocí příkazy `vgexted` je možné přidat do stávající skupiny další fyzický svazek. Například, pokud do systému přidáme další disk, pokud možno hot-swap, a vytvoříme na něm fyzický svazek, lze jej následujícím příkazem přidat do skupiny.

```
# vgextend -v volg01 /dev/sda1
vgextend - doing automatic backup of volume group "volg01"
vgextend - volume group "volg01" was successfully extended by phisical volume:
vgextend - /dev/sda1
```

Opakem k zvětšení skupiny je odstranění některého jejího fyzického svazku. Tato operace ale nemůže být provedena tak přímočaře, protože některé extenty vyjímaného fyzického svazku už by mohly být používány některým logickým svazkem. Vrátíme-li se k obrázku, potom nelze druhý fyzický svazek vyjmout ani za předpokladu, že zrušíme druhý logický svazek, protože jeden jeho extent je používán prvním svazkem. Tato problematika bude probírána dále. Pokud jsme si ale jisti a výpis programu `vgdisplay` nás o tom přesvědčil, že nic takového nehrozí, můžeme vyjmout fyzický svazek následovně.

```
# vgreduce -v volg01 /dev/sda1
vgreduce - doing automatic backup of volume group "volg01"
vgreduce - volume group "volg01" was successfully reduced by phisical volume:
vgreduce - /dev/sda1
```

Pokud není fyzický svazek schopen redukce, to jest obsahuje některé alokované extenty, systém jej neuvolní. Zároveň bych chtěl upozornit, že všechny operace, jak už tomu v Linuxu bývá, jsou *bez ptaní*. Zvlášť nebezpečné je zaměnit si příkazy `vgreduce` a `vgremove`. Druhý z nich bez milosti zlikviduje celou skupinu svazků. Což je svým způsobem také redukce, ale bývá zpravidla doprovázena okamžitou výpovědí pracovní smlouvy.

Jsou-li ve skupině svazků volné extenty, můžeme je přiřadit k některému *logickému svazku*. Analogicky jako v předcházejícím případě to ale ještě nepovede ke zvětšení *souborového systému*. Lze si to představit tak, že za konec souborového systému se ještě přidá místo, ale souborový systém jej sám o sobě neobsadí. On v podstatě neví, že je za ním ještě nějaké místo.

Před samotným zvětšením se doporučuje odmountovat souborový systém, který je na zvětšovaném logickém svazku. Sice to teoreticky není nutné, ale před manipulací se souborovým systémem je dobré dodatečně otestovat jeho integritu, což samo o sobě v mnoha případech nelze na namountovaném souborovém systému provést. Dobré řešení je rovněž remountovat souborový systém pouze pro čtení příkazem `mount -o ro,remount /dev/volg01/blah`. V případě kritický aplikací, kdy data musejí být stále dostupná, se samozřejmě může udělat vše „za běhu“. Zvětšení logického svazku lze provést třeba takto.

```
# lvextend -L+100MB /dev/volg01/blah
lvextend - extending logical volume "/dev/volg01/blah" to 900 MB
lvextend - doing automatic backup of volume group "volg01"
lvextend - logical volume "/dev/volg01/blah" successfully extended
```

Po zvětšení svazku následuje zvětšení souborového systému. Pro souborový systém EXT2, který je nyní asi masivně nejpoužívanější, existuje utilita `resize2fs`, pomocí které lze souborový systém zvětšit, nebo zmenšit. Souborový systém ReiserFS umožňuje změnit velikost souborového systému pomocí argumentu `resize`, určujícímu počet bloků, na které se má souborový systém natáhnout nebo zmenšit. Stejně tak součástí balíku programů pro ReiserFS je i program umožňující manipulovat s velikostí souborového systému za běhu.

2.5 Vyjmutí disku

Další výhoda LVM, která je k nezaplacení. Jestliže se vám zdála architektura LVM příliš složitá a intuitivně jsme cítili, že jedna vrstva lze odstranit bez újmy na funkčnosti, tak doposud jste měli pravdu. Velmi často se ale stává, že se jeden z disků v poli začne kazit nebo nedej bože, někdo koupí nové diskové pole a my jsme postaveni před úkol „jednoduše provést migraci dat“.

Ať už je taková migrace vynucena chybným médiem nebo ne, narážíme opět na problém, jak odstranit fyzický svazek ze skupiny svazků. Navíc ale tak, aby extenty, které obsahuje, byly zachovány. Jsou-li zachovány tyto extenty, jsou automaticky neporušeny i logické svazky. Řečeno jinak, na úrovni souborových systémů by jádro takovou výměnu nemělo poznat.

Pomocí příkazu `pvmove` je možné přesunout extenty z jednoho fyzického svazku na druhý, je možné přímo určit na který fyzický svazek je třeba extenty přesunout. Viz příklad.

```
# pvmove /dev/hda1
pvmove - moving physical extents in active volume group "volg01"
pvmove - WARNING: moving of active logical volumes may cause data loss
pvmove - doing automatic backup of volume group "volg01"
pvmove - 430 extents of physical volume "/dev/hda1" successfully moved
```

No, takže když nám někdo věnuje nové diskové pole, tak jej za běhu hodíme do počítače, pokud je to možné a vytvoříme na něm fyzický svazek. Ten přidáme do existující skupiny svazků. Ta bude kromě nového fyzického svazku obsahovat i ty, které budeme chtít vyjmout. Provedeme `pvmove` a po úspěšném zkopírování extentů vyřadíme staré fyzické svazky ze servery pomocí příkazu `pvreduce`. Staré diskové pole za běhu vyhodíme. Uživatelé nic nepoznali. Všechno proběhlo plně transparentně. Naprosto stejným způsobem lze postupovat v případě vadného disku.

2.6 Přesuny disků mezi počítači

S tím, jak se některé úkony výrazně zjednodušily, jiné se na druhou stranu ztížily. Při použití LVM není možné jen tak přesunout disk s LVM na jiný počítač, protože informace o svazcích jsou umístěny i v souborech v adresáři `/etc/lvmconf`. Pokud chceme disk s LVM fyzicky přenést na jiný počítač, musíme nejprve skupiny svazků odhlásit od jádra a zazálohovat informace o svazcích pomocí příkazu `vgexport`.

```
# vgchange -a n volg01
# vgexport volg01
```

Na druhém počítači potom aktivujeme svazky následujícími příkazy.

```
# vgimport jina_volg /dev/hda2 /dev/hdc1
# vgchange -a n jina_volg
```

Jména skupin svazků nemusejí být na počítačích stejná.

2.7 Další možnosti LVM

LVM umožňuje ve skupině svazků shlukovat více fyzických svazků. Jednotlivé logické svazky potom mohou mít extenty umístěny na navzájem různých svazcích. Z důvodů výkonu je výhodné, aby nebyly fyzické svazky „zřetězovány“, nýbrž „prokládány“. Tomuto mechanismu se říká *stripping*, neboli *RAID Level 0*. Zjednodušeně řečeno jde o to, že bloky, které jsou za sebou jsou střídavě hledány na několika discích, tím se de facto násobí rychlost transferu dat. Ovšem pouze za předpokladu, že to sběrnice stíhá. Například RAID 0 na sekundárním IDE kontroleru by jedině odrovnalo oba disky.

LVM disponuje mechanismem *native stripe*, při vytváření logického svazku je možné určit, na kolika fyzických svazcích se mají alokovat extenty a rovněž lze specifikovat granularitu jednotlivých „proužků“. Při použití těchto parametrů se vytvoří nový logický sazek, který bude používat nativní LVM stripes.

```
# lvcreate -L 20M -i 2 -I 64 -n new_lvolg volg01
lvcreate - rounding 20480 KB to stripe boundary size 24576 KB / 6 PE
lvcreate - doing automatic backup of "volg01"
lvcreate - logical volume "/dev/volg01/new_lvolg" successfully created
```

V současné době se pracuje i na konceptu zrcadlení, RAID 1. Do budoucna je dost dobře možné, že projekty softwarového RAIDu a LVM se spojí v jeden.

3 Specialisované souborové systémy

Linux disponuje množstvím specialisovaných souborových systémů, dva z nich, které jsou vhodné nejen při vytváření zaváděcích a záchranných disket, jsou dále popsány. Souborový systém `ROMFS` najde uplatnění i v malých zařízeních, které disponují malou EEPROM pamětí, jako jsou PDA, mobilní telefony a podobně.

3.1 Souborový systém ROMFS

Souborový systém `ROMFS` je velmi štíhlý souborový systém pouze pro čtení, původně vyvinutý pro inicialisační a bootovací diskety, které jsou omezené svou kapacitou. V dnešní době není úplně jednoduché vyrobit zaváděcí disketu, která by obsahovala jednak jádro a jednak komprimovaný souborový systém. Jakákoliv možnost, snížit velikost jádra, je proto velmi významná.

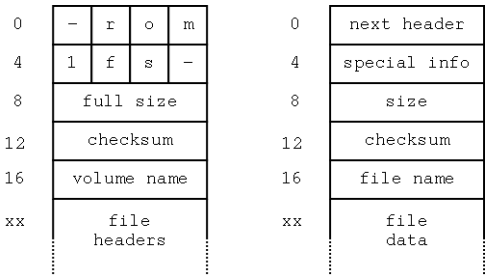
Souborový systém `ROMFS` je možné vytvořit externím programem, zhotovený obraz je možné umístit na bootovací medium nebo do paměti typu PROM, EEPROM a podobně. K výrobě souborového systému je tedy potřeba mít již připravenou strukturu adresářů a souborů. Do vytvořeného souborového systému již není možné žádným způsobem zapisovat, což ale vzhledem k jeho nasazení ani nebylo účelem.

Pro porovnání, implementace velmi malých souborových systémů, jako jsou Minix a XIAFS, zabírá v jádře téměř 20 kB. Implementace souborového systému EXT2, která se doposud pro zaváděcí diskety používala, zabírá téměř 60 kB. ReiserFS zabírá přes 140 kB. Implementace ROMFS se vejde do jednoho bloku, je menší než 4096 B. I když se nám to z pohledu bootovací diskety může zdát málo, z pohledu náramkových hodinek společnosti IBM, které v sobě obsahují implementaci Linuxu, je to podstatný rozdíl.

Ačkoliv je implementace souborového systému velmi malé, zaručuje existenci adresářové struktury, umožňuje vytvářet speciální soubory pro bloková a znaková zařízení, sockety, roury, linky a podobně. V následujícím textu je popsána struktura `ROMFS` a z ní vyplývající omezení.

3.1.1 Vnitřní struktura

Souborový systém `ROMFS` funguje na blokovém zařízení, jak by se dalo očekávat, jeho struktura je velmi jednoduchá. Každá dosažitelná struktura začíná na adrese dělitelné 16, to je z důvodů rychlosti přístupu. Každý soubor zabere přinejmenším 32 B — to odpovídá prázdnému souboru, jehož jméno je kratší než 16 znaků. Maximální režie pro každý neprázdný soubor je právě 16 bytové zarovnávání pro název souboru a jeho obsah, v nejhorším případě činí 45 bytů na soubor. Následující obrázky ukazují design souborového systému.



Obrázek vlevo zahrnuje strukturu celého souborového systému, obrázek vpravo representuje jeden soubor. Viz dále. Prvních 8 bytů slouží k identifikaci souborového systému, je v nich umístěn řetězec „`-rom1fs-`“. Pro souborový systém `ROMFS` je zvolena konvence, že každá multibytová hodnota, například 32-bitový integer je kódována v *big endian*. Bezprostředně za identifikátorem souborového systému je celočíselná hodnota *full size*, která representuje počet bytů, které přístupné od počátku souborového systému. Další čtyřbytový integer *checksum* je kontrolní součet prvních 512 bytů souborového systému, jde o prostý součet. Tento algoritmus je sice triviální, ale pro dané nasazení dostačující. A potřeba kontroly integrity dat na médiu pouze pro čtení není tak vysoká.

Další položku tvoří název svazku *volume name*. Jedná se o nulou zakončený řetězec. Řetězec může mít libovolnou délku, ale vzhledem k tomu, že všechny jednotky začínají na adrese dělitelné 16, musí být místo pro řetězec doplněno případně více nulami tak, aby respektovalo tuto hranici. Za jménem svazku jsou obsaženy jednotlivé hlavičky souborů, jejich počáteční adresa není pevně dána a záleží právě na délce *volume name*.

Samotnou organisaci hlaviček souborů si lze představit jako linearisovaný strom, který je umístěn na disku. Z praktických a výkonových důvodů je strom *linearisován do šířky*, nikoliv do hloubky. Při vstupu do adresáře, jsou všechny jeho soubory uloženy na médiu sekvenčně za sebou, takže čtení jejích informací i jejich dat je velmi pružné.

Hlavička souborů začíná čtyřbytovým odkazem na hlavičku dalšího souboru, pokud je hodnota ukazatele nulová, další soubor neexistuje, bráno samozřejmě v rámci jednoho adresáře, obecně mohou ještě za konkrétním záznamem existovat další. Jelikož jsou všechny hlavičky zarovnány na adresu dělitelnou 16, spodní 4 bity adresu zůstávají nevyužity. Ve spodních 4 bitech může být uložena dodatečná informace, bity 0..2 určují typ souboru, význam jednotlivých hodnot je následující.

0	*hard link*	4	*blokové zařízení*
1	*adresář*	5	*znakové zařízení*
2	*regulární soubor*	6	*socket*
3	*symbolický link*	7	*roura*

Poslední bit je příznak spustitelnosti souboru. V UNIXu jsou klasicky přístupová práva charakterisována 44 bity nebo 76 bity, které dostačují k určení vlastníka, skupiny a přístupových práv. U souborového systému `ROMFS` se vlastníci souboru nerozlišují, všechny soubory jsou vlastněny uživatelem 0 a skupinou 0, což je `root`. Přístupová práva jsou buďto `0644` pro soubory, kter0 nejsou spustitelné, nebo `0755` pro soubory, které jsou spustitelné. V případě speciálních zařízení má daná práva na soubor z bezpečnostních důvodů pouze vlastník.

Další položkou v klavičče souboru je *special info*. Jde o informaci, jejíž kontext je závislý na typu souboru. U regulárního souboru, symbolického linku, socketu a roury je tato položka nevyužitá. *Adresář* obsahuje jako

speciální informaci ukazatel na hlavičku prvního souboru, stejně tak *hard link* obsahuje odkaz na cílovou hlavičku souboru. Speciální zařízení, bloková a znaková, zde mají uloženou informaci o jejich major/minor identifikačním čísle.

Položka *size* definuje velikost dat souboru v bytech. Komtrolní součet *checksum* je součtem prvních 512 bytů souboru. Jméno souboru je opět zakončené nulou a doplněno na adresu dělitelnou 16. Stejně tak i vlastní data souboru. Symbolický link je realisován tak, že cíl je identifikován řetězcem v datová oblasti souboru.

3.2 Souborový systém RAMFS

Při vyrábění jednodisketových distribucí, typicky například záchranných disket a podobně je dobré mít k dispozici i dočasné místo na zápis. Malý souborový systém `ROMFS` je pouze pro čtení, ale výborně se doplňuje s malým systémem `RAMFS`, který udržuje všechny své záznamy trvale v paměti a po odmountování data nenávratně zmizí.

Implmentace `RAMFS` má 8 KB zdrojových kódů, implementace `ROMFS` 13 KB, takže se v podstatě jedná o ještě štíhlejší souborový systém. Vzhledem k vlastnostem UNIXového virtuálního souborového systému, `RAMFS` stačí pouze manipulovat s inody v paměti a všechny požadavky na zápis ignorovat. Tato myšlenka vede v velmi jednoduché implementaci souborového systému. Oproti stávajícím RAM-diskům, které mají vždy pevně danou velikost, `RAMFS` se může natahovat a smršťovat plně dle potřeby. Na třistařádkový program, klobouk dolů.

Linux obsahuje virtuální vrstvu, která vytváří abstrakci nad všemi souborovými systémy, je to *VFS — Virtual File System*. VFS umožňuje přistupovat k adresářová struktuře a zcela odstiňuje zbytek jádra od souborové vrstvy. Jinak řečeno, proces na funkčnosti nepozná, zda-li je na EXT2 nebo na EXT3, samozřejmě za předpokladu, že oba souborové systémy implementují tytéž funkce. VFS si o každém namountovaném souborovém systému udržuje informace v *superbloku*. Zde jsou adresu funkcí, které implementují jednotlivá volání jádra, když si jádro například vyžádá soubor, zavolá se příslušná funkce, která tento problém řeší na úrovni daného souborového systému. Informace o samotných souborech jsou uloženy ve VFS *inodech*, ty nesou nezbytné informace o souboru, včetně práv, časů přístupu, modifikace, aktualisace a podobně.

Implementace `RAMFS` je z tohoto pohledu přímočará, protože veškeré funkce, sloužící pro zápis, například `fsync` nebo `block_write` jsou realisovány tak, že stránkám nastaví příznak `dirty`. Tím pádem je jádro nemůže uvolnit a zůstávají alokovány v paměti. Zápis na disk se neuskutečňuje. Čtení souborů je triviální, protože všechny stránky jsou umístěny trvale v paměti. Obdobným způsobem bylo rovněž dosaženo toho, že souborový systém „roste dne požadavků“ uživatele, tedy zabírá právě tolik, vyjma režie, kolik jsme na něj uložili.

Máme-li zkompilovanou podporu `RAMFS` v jádře systému, lze jej namountovat pomocí následujícího příkazu.

```
# mount -t ramfs ramfs /misto
```

Po odmountování data zmizí. Tento souborový systém bude mít široké využití v takzvaných Native CD Distributions, jde v podstatě o CD s Linuxem, které obsahuje nativní souborový systém a po nabootování může uživatel pracovat se základním softwarem a grafickým rozhraním. Právě na takovém médiu pouze pro čtení je potřebné mít zapisovetelné alespoň některé části souborového systému, třeba adresář `/tmp` pro vytvoření zámků a podobně. V tomto směru zajímavý projekt je Bootable Business Card, který obsahuje malou distribuci Linuxu odvozenou z distribuce Debian GNU/Linux, která se vejde na CD nosič velikosti kreditní karty.