A Logical Volume Manager

Vajon hányan ültek már büszkén az újonnan beüzemelt szerverük előtt azon gondolkodva, hogy ezzel a rendszerrel bizony évekig semmi komolyabb gondja nem lesz. Három vagy még több merevlemez, örökké 80% alatt terhelt fájlrendszerek...

Aztán telnek-múlnak a hónapok, szabad hely pedig egyre csak fogy, és rendszergazdánk lassan már elgondolkodik, hogy mihez is kezdjen a szervere csordulásig telt fájlrendszereivel, és hogyan is tudná felhasználni az alig használtakat. Mennyivel könnyebb lenne az élet, ha egy eredetileg túlméretezett partíció rovására növelhetnénk a helyhiánnyal küszködő partíciónkat, vagy egyszerűen csak kibővítenénk egy újonnan beüzemelt merevlemezzel a megterhelt fájlrendszerünket, és mindezt persze újra telepítés, megkockáztatom: újraindítás nélkül... Röviden erre (is) jó az *LVM*.

Unix örökség

Az *LVM* (*Logical Volume Manager*) ötlete még a jó öreg *Unix* rendszereken valósult meg elsőként (azóta szinte az összes operációs rendszer támogatja). A kilencvenes évek elején már a *HP-UX* része volt, később az *OS/2*-ben illetve a *Solarisban* (*Solaris Volume Manager*) is megvalósult. *AIX*, **BSD* mellett természetesen *Linuxon* is elterjedt eszközzé vált.

Az LVM az mi az?...

Azt azért tudni kell, hogy nem csak *Linux*/*Un*x* környezetben használatos az *LVM*. Lefogadom, sokan találkoztak már az *LDM* (*Logical Disk Manager*) fogalommal, amit a *Microsoft* a *Windows* 2000-ben vezetett be. Vagyis a „*Basic*", *DOS*-ból örökölt lemezkezelés mellett megjelent a dinamikus lemezkezelés, amit eredetileg egy *VERITAS* nevű cég fejlesztett ki Unix alá, majd ültette át *Windows NT*-re (későbbiekben licencelte a *Microsoft* a 2000-es változathoz). Funkcióiban találhatunk hasonlóságot a *Linux* alatt fellelhető *LVM* és az *LDM* között, de megvalósításban teljesen különbözőek (igaz, az *LDM* is a fizikai partíciókon tárol minden leíró információt (*metadata*) a logikai kötetről, sőt ugyanúgy *UUID* (*Universally Unique Identifier*) alapján azonosítja őket). Vicces, de meg kell jegyeznem, hogy az *LDM* óta képes a *Windows* is „mountolni" partíciókat. Ami késik nem múlik.
Térjünk vissza az *LVM*-hez. Az első *LVM* meghajtó a 2.3.47-es kernelben kapott helyet, ezért az ebből a fából kiemelt 2.4.x sorozat már aktívan tartalmazta. A 2.6-os kernelfa már az *LVM2*-t (is) támogatja, melyet a kernelbe bekerült *device-mapper* valósít meg.
LVM segítségével a gépünkben található háttér tárakat akár egyben, egy logikai háttértárként használhatjuk, melyet utólag particionálhatunk igényeinknek megfelelően. Hatalmas előnye, hogy akár *RAID* tömbre is felépíthető az *LVM*, sőt több *RAID* tömböt is összefoghatunk egy logikai eszközzé. Új merevlemez hozzáadásakor egyszerűen csak megnöveljük partíciónk méretét, vagy akár teljes lemezterületeket mozgathatunk át rá.

Alapok

A fentebb említett előnyeit az *LVM*-nek nem túl nagy ördöngösség kihasználni, igaz, nem árt néhány alap fogalmat tisztázni, mielőtt az „építkezést" elkezdenénk.

- *PV (Physical Volume)*: Ez az alap és egyben legkisebb egysége az *LVM*-nek. A *PV* nem más, mint egy inicializált *LVM* típusú partíció. Itt még nincs logikai kapcsolat felépítve a fizikai kötetek között.
- *VG (Volume Group)*: *PV*-ket összefoglaló, névvel ellátott logikai kötetcsoport. Általában egy *VG* több, akár különböző háttér tárakon elhelyezkedő *PV*-ből épül fel.
- *LV (Logical Volume)*: Logikai kötet, tulajdonképpen egy partíció az *LVM* felett. Létező *VG*-re épül. Egy *VG*-re több *LV* is létrehozható (képzeljük el úgy, hogy a *VG* a fizikai lemez, az *LV*-k pedig partíciók).

A cikkben az *1. ábrán* látható *LVM* szervezetet fogjuk lépésről lépésre létrehozni, vagyis kezdésként négy *PV*-ből létrehozunk egy *VG*-t, majd azt kettéosztva fogunk kapni két darab *LV*-t. Ez hát a cél...

Építkezés lépésről lépésre

Ez az a pont ahol meg kell jegyeznem, hogy az itt leírtakat éles rendszerben *ne* alkalmazzuk. Csak kipróbált, agyontesztelt *LVM* konfigurációt állítsunk csatasorba, sok-sok kellemetlen adatvesztéstől mentesülve ezáltal.
Első lépés, elővesszük kedvenc partíció kezelő programunkat, és elkezdünk *Linux LVM* (0x8E) típusú partíciókat gyártani.
Ha ezzel megvagyunk, jöhet a partíciók inicializálása a `pvcreate` segítségével:

```
# pvcreate /dev/hda[5-8]
  Physical volume "/dev/hda5"
  ➥successfully created
  Physical volume "/dev/hda6"
  ➥successfully created
  Physical volume "/dev/hda7"
  ➥successfully created
  Physical volume "/dev/hda8"
  ➥successfully created
```

Jó esetben hiba nélkül létrejött a négy *PV*, jöhet a csoportba szervezés a

```
# vgcreate vgteszt /dev/hda
# [5-7]
  Volume group "vgteszt"
  ➥successfully created
```

paranccsal.
A *„VG Size”* értékéből látható, hogy maximum 9,31Gb méretű *LV*-t hozhatunk létre. Készítsünk egy 2 GB *LV*-t:

```
# lvcreate -L2048 -nlvkotet01
# vgteszt
  Logical volume "lvkotet01"
  ➥created
```

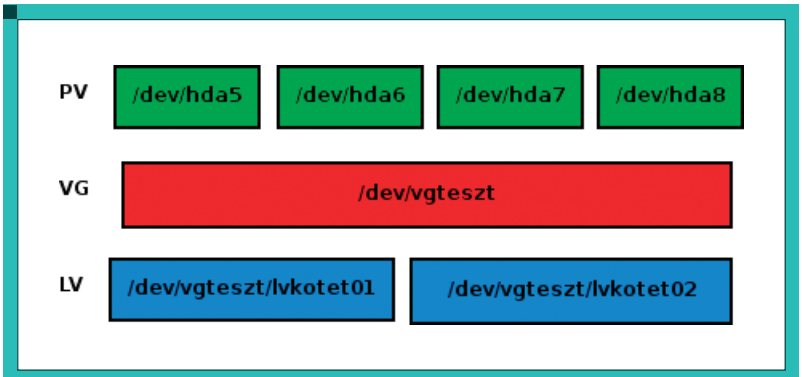
A maradék 7,3 GB *VG* kapacitáson pedig készítsük el második *LV* kötetet:

```
# lvcreate -L7,3G -nlvkotet02
# vgteszt
```

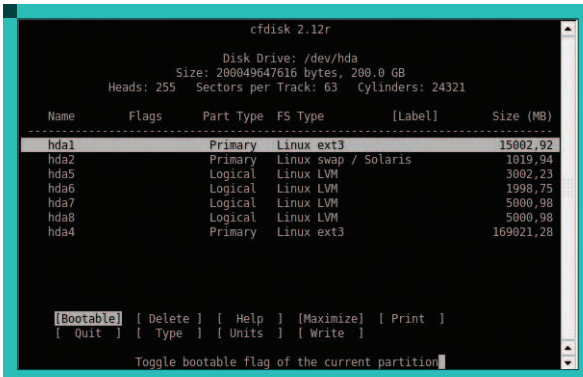
Nincs más hátra, már csak használatba kell venni újdonsült logikai köteteinket, formázzuk le őket kedvenc fájlrendszerünkkel, mondjuk használjunk *ext3*-at:

```
# mkfs.ext3 /dev/vgteszt/
# lvkotet01
# mkfs.ext3 /dev/vgteszt/
# lvkotet02
```

Gondolom, mindenkinek feltűnt, hogy a *VG* létrehozásakor „véletlenül” kihagytam a negyedik *PV*-t, a `/dev/hda8`-at. A megoldásnak az *LV*-k és a *VG* törlése, majd új, mind a négy *PV*-t tartalmazó csoport létrehozása, ami ugyan hatékony lehet üres kötetnél, de mégis van ettől egy egyszerűbb *LVM* eszköz, a `vgextend`. Ezzel az eszközzel egy már meglévő kötetcsoporthoz adhatunk hozzá újabb fizikai kötetet, ezzel is bővítve kapacitását. Szóval adjuk hozzá a hiányzó *PV*-t a *„vgteszt”* csoporthoz:



1. ábra PV-VG-LV viszony. Nem is annyira bonyolult...



2. ábra Igaz, egy merevlemezen, de elkészült a négy darab „nyers” PV kötet.

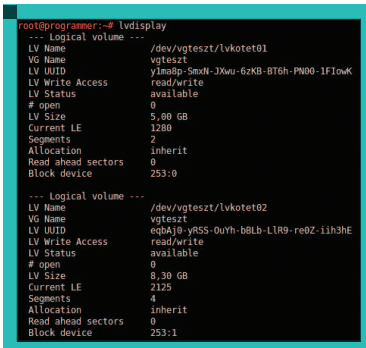
```
root@programmer:~# vgdisplay
  --- Volume group ---
  VG Name               vgteszt
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               9,31 GB
  PE Size               4,00 MB
  Total PE              2383
  Alloc PE / Size       0 / 0
  Free  PE / Size       2383 / 9,31 GB
  VG UUID               4eORU6-6rE8-S0PM-BNhX-O6FK-8qOA-EvdoXZ
```

3. ábra A létrehozott vgteszt nevű kötetcsoportról a vgdisplay parancs által kaphatunk némi információt.

```
# vgextend vgteszt /dev/hda8
  Volume group "vgteszt"
  ➥successfully extended
```

Ezek után meg kell növelnünk a logikai kötetek méreteit is. Azt szeretnénk, ha a `lvkotet01` mérete 2 GB-ról 5 GB-ra növekedne, míg az `lvkotet02` méretét 1 GB-tal szeretnénk növelni:

```
# lvextend -L5G
# /dev/vgteszt/lvkotet01
  Extending logical volume
  ➥lvkotet01 to 5,00 GB
  Logical volume lvkotet01
  ➥successfully resized
# lvextend -L+1G /dev/
# vgteszt/
# lvkotet02
```



■ *4. ábra* Az lvdisplay segítségével ellenőrizzük munkánkat.

```
  Extending logical volume
  ➥lvkotet02 to 8,30 GB
  Logical volume lvkotet02
  ➥successfully resized
```

Már majdnem kész vagyunk, de a fájlrendszerek átméretezése még hátravan. Ha a `resize2fs`-nek nem adjuk meg paraméterként az új méretet, akkor a fájlrendszert magában foglaló partíció maximális méretéhez igazít:

```
# resize2fs /dev/vgteszt/
# lvkotet01 && resize2fs
/dev/vgteszt/lvkotet02
```

Elvileg kész. A

```
# mount /dev/vgteszt/lvkotet0x/
# elérési/út
```

paranccsal már használatba is vehetjük *LV* kötetünket.

Rombolunk, majd újjáépítünk

Egy szép nap észrevesszük, hogy a `smartd` csúnyákat írogat log fájlunkba, és rájövünk, hogy az egyik, *LVM* alatt zakatoló merevlemez bizony-bizony a végét járja. Persze nem ijedünk meg, elegáns mozdulattal elővesszük tartalék meghajtónkat a fiókból (ha nincs, akkor eszeveszetten lavírozunk a legközelebbi számítástechnikai boltba), és nemes egyszerűséggel beüzemeljük haldokló szerverünkbe. Jön a már megismert `pvcreate`, majd újdonsült fizikai kötetünket a

```
# vgextend vgteszt /dev/hda9
  Volume group "vgteszt"
  ➥successfully extended
```

paranccsal hozzáfűzzük létező *LVM* csoportunkhoz. Most már csak az adatokat kell átmozgatunk a beteg kötetről az újra:

```
# pvmove /dev/hda6 /dev/hda9
  /dev/hda6: Moved: 8,2%
  /dev/hda6: Moved: 17,0%
  /dev/hda6: Moved: 25,4%
  ...
  /dev/hda6: Moved: 96,7%
  /dev/hda6: Moved: 100,0%
```

A `pvmove` nem kapkodja el, PV méretétől függően akár kávézni is elmehetünk. Ha a parancs futtatásakor az

```
  Insufficient free space: 715
  ➥extents needed, but only 24
  ➥available
  Unable to allocate temporary
  ➥LV for pvmove.
```

hibaüzenethez hasonlót kaptunk, akkor biztosak lehetünk benne, hogy a régi *PV extent*-jei nem férnek el az újonnan csatolt fizikai köteten. Az is előfordulhat, hogy a `pvmove` parancsra a

```
mirror: Required device-mapper
➥target(s) not detected in
➥your kernel
```

hibaüzenet jelenik meg. Ilyenkor jól jön, ha tudjuk, hogy a `dm-mirror` modulunk nincs betöltve. Ezen segít a

```
#modprobe dm-mirror
```

parancs. Már majdnem készen vagyunk, már csak a jó öreg betegeskedő PV-től kell megszabadulnunk:

```
# vgreduce vgteszt /dev/hda6
  Removed "/dev/hda6" from
  ➥volume group "vgteszt"
```

Ha minden jól ment, akkor merevlemez ki, selejtezés (*torrentek* hazahurcolására meg jó lehet).

Átszabjuk

Természetesen nem csak *LVM* bővítésére, hanem átméretezésére is igény lehet. Első lépés, fájlrendszereket lecsatoljuk (nekem az `/mnt/lv0x` mappába lettek csatolva):

```
#umount /mnt/lv0[1-2]
```

Jöhet a fájlrendszer csökkentése, melyre szintén a `resize2fs` parancsot használhatjuk. Mondanom sem kell, az említett példák *ext2/3* fájlrendszerre vonatkoznak, *reiserfs*, *xfs* esetén már parancsokat kell használnunk (főleg *xfs* esetén, mivel ezt a típusú fájlrendszert a *jfs*-hez hasonlóan nem lehet csökkenteni).

```
#resize2fs /dev/vgteszt/
#lvkotet01 2G
resize2fs 1.39 (29-May-2006)
Resizing the filesystem on
➥/dev/vgteszt/lvkotet01 to
➥524288 (4k) blocks.
The filesystem on /dev/
➥vgteszt/lvkotet01 is now
➥524288 blocks long.
```

Most jön a logikai kötet csökkentése/növelése a `lvreduce/lvextend` paranccsal:

```
# lvreduce -L2G /dev/
# vgteszt/lvkotet01
  WARNING: Reducing active
  ➥logical volume to 2,00 GB
  THIS MAY DESTROY YOUR DATA
  ➥(filesystem etc.)
Do you really want to reduce
➥lvkotet01? [y/n]: y
  Reducing logical volume
  ➥lvkotet01 to 2,00 GB
  Logical volume lvkotet01
  ➥successfully resized

# lvrextend -L+2G /dev/
# vgteszt/lvkotet02
  Extending logical volume
  ➥lvkotet02 to 10,30 GB
  Logical volume lvkotet02
  ➥successfully resized
```

Az `lvreduce` figyelmeztet, hogy a művelet adatvesztéssel járhat. Ezt vegyük komolyan, de mi nyugodtan folytassuk a redukálást, hiszen előtte már lecsökkentettük a fájlrendszer méretét. A `lvkotet02` fájlrendszert szintén a `resize2fs` segítségével növeljük. A biztonság kedvéért érdemes a fájlrendszereinket ellenőrizni:

```
# fsck.ext3 -f /dev/
# vgteszt/lvkotet01 ;
# fsck.ext3
# -f /dev/vgteszt/lvkotet02
```

`pvremove`	Használaton kívüli (egyetlen csoporthoz sem tartozó) fizikai köteteket tudunk "LVM-mentesíteni".
`pvresize`	Segítségével VG-hez csatolt, sőt aktív LV alatt működő fizikai kötetet tudunk átméretezni. Leginkább PV-k csökkentésére érdemes használni, bár én ha nem muszáj, nem nyúlnék éles rendszerben ehhez az eszközhöz.
`pvs`	Formázott riportot kapunk fizikai köteteinkről. A vgs parancs nagyon hasonlóan működik, használatával a VG-ről kapunk információt, míg a lvs az LV-k adatairól ad hasonló riportot.
`pvscan`	Minden partíciót megvizsgál, PV-k után kutatva, a vgscan kötetcsoportokat keres, míg a lvscan inaktív logikai köteteket.
`vgcfgbackup`	A kötetcsoportok leíró adatairól (metadata) készít másolatot szöveges fájlba általában az /etc/lvm/backup (amit a -f kapcsolóval megváltoztathatunk) mappába. Félreértések elkerülése végett, ez nem a felhasználói adatokat archiválja! Szorosan ide kapcsolódik a vgcfgrestore parancs, mely a mentett szöveges állományból állítja vissza a csoport metaadatait.
`vgchange`	Kötetcsoportok néhány tulajdonságát befolyásolhatjuk. A -a kapcsolóval a csoport elérhetőségét engedélyezhetjük/tilthatjuk, -A kapcsolóval a metadata automatikus mentését befolyásolhatjuk illetve a csoport átméretezhetőségét és a maximum LV számot is megadhatjuk.
`vgconvert`	Kötetcsoport leíró adatait konvertálhatjuk át más formátumúvá (nagyon hasznos, ha LVM1 formátumú VG-t akarunk LVM2-ként használni).
`vgmerge`	Két létező kötetcsoportot tudunk vele összekapcsolni. A csatolandó kötetcsoportnak mindenképp inaktívnak kell lennie (vgchange -a n). Hasznos lehet a -t kapcsoló, mivel így a folyamatot valós csatolás nélkül tesztelni. Ellentéte a vgsplit, ami két részre oszt egy VG-t (ne felejtsük, a VG-n található LV-t nem lehet "kettévágni").
`vgck`	A kötetcsoportok metaadatainak konzisztenciáját ellenőrizni. Örüljünk, ha nincs a programnak semmilyen kimenete.
`vgmknodes`	Ellenőrzi, hogy az aktív csoportoknak megfelelő bejegyzések léteznek-e a /dev-ben. Ha nem, létrehozza őket (nem jelent jót, ha használnunk kell).
`lvchange`	Logikai kötet aktiválásán/leállításán kívül alapvető jellemzőket állíthatunk vele.
`lvmdump`	Hasznos debug eszköz. Minden hasznos és haszontalan információt összeszed rendszerünkről (futó folyamatok, eszközlista, LVM konfiguráció stb.).

Figyelem, felvétel indul!

Rendkívül hasznos, mégis alul dokumentált funkciója az ***LVM***-nek a ***snapshot*** készítés. Segítségével a logikai köteten elhelyezkedő fájlrendszerről készíthetünk egy „pillanatfelvételt”, melyet akár archiválhatunk, természetesen az adatok konzisztenciája megőrzése mellett. Kernelünk nagyon egyszerű, de nagyszerű módon oldja meg ezt a szolgáltatást. A ***snapshot LV*** létrehozásakor igazából csak egy „tükörkép” jön létre (semmi duplikáció) az archiválandó logikai kötetről. A ***snapshot*** élete alatt bármilyen tranzakció hajtódik végre a forráson, akkor az eredeti adatok átkerülnek a ***snapshot*** kötetre, és csak ezután hagyja a kernel a módosult adatok kiírását. Így a forrás logikai kötet illetve a ***snapshot*** logikai kötet adatait összevetve a ***snapshot*** készítés időpontjában létezett állapotot kapjuk (vagyis a két ***LV*** tartalmából kernelünk készíti el a „pillanatfelvétel” tartalmát, mivel ő pontosan tudja, mely blokkok tartalma módosult). Természetesen ahhoz, hogy a ***snapshot LV***-t létrehozzuk, szabad kapacitásnak is kell lennie a ***VG***-n. Próbaképp csináljunk egy 50Mb méretű „felvételt” az `lvkotet01`-ről:

```
# lvcreate -L50M -s -n
# lvpillfelv /dev/vgteszt/
# lvkotet01
  Logical volume "lvpillfelv"
  ➥created
```

Ha most felcsatoljuk az újonnan létrejött logikai kötetet (`/dev/vgteszt/lvpillfelv`), akkor az `lvkotet01` tartalmához hasonlót találunk rajta. A ***snapshot*** méretére figyeljünk oda. Bizonyos módosításszám után könnyen megtelhet, oda a „felvétel”. Az optimális méret függ a forrás kötet nagyságáról, fájlok átlagos méretétől, felhasználók és az általuk indított tranzakciók számától. Ideális, igaz helypazarló megoldás, ha a ***snapshot*** mérete megegyezik a forrás ***LV*** méretével, így biztos, hogy soha nem fogunk abból kifutni a mentés alatt. Pazarló, de egyszerű napi mentést is képesek vagyunk létrehozni az ***LVM snapshot*** funkciójával, hiszen nincs más dolgunk, mint minden nap egy új „pillanatfelvétel”-t készíteni heti forgásban (vagyis hétfőn töröljük az előző hétfőt, majd létrehozzuk az újat), amit ***cron*** démon segítségével éjszakára is ütemezhetjük.

Ha költözik az LVM

Egyszer eljön a pillanat, mikor egy működő ***LVM*** rendszert valamilyen úton-módon át kellene költöztetni egy másik gépbe, esetleg teljesen különböző disztribúció alá. A ***PV***-k merevlemezeit szépen átköltöztetjük az új gépbe, és eljön a varázslás ideje, `pvscan`, `vgscan` stb-stb...

Ekkor jön egy újabb hasznos funkció, az export-import. A régi gépen kiadjuk a

```
# vgchange -a n && vgexport -a
```

parancsokat, melyek hatására az összes *VG* szépen megáll, a `vgexport` hatására pedig bekerül az `EXPORTED` bejegyzés az `/etc/lvm/backup/vgteszt` fájlba (ide is, meg a kötetleíróba is), magyarul a logikai kötet leválasztható rendszerükről. Nincs más dolgunk, mint a merevlemezeket átszerelni új gépünkbe, majd kiadni a

```
# vgimport -a && vgchange -a y
```

parancsokat.

Vegyes-vágott, ami kimaradt

Az *LVM* jó néhány tulajdonságáról illetve a kezelésére szánt bináris csomagban megtalálható néhány segédprogramról nem beszéltünk még, ezért jöjjön néhány (általam hasznosnak vélt) funkció.
`pvchange`: Kevés dokumentációban említik, pedig hasznos programocska. Segítségével paraméterként megadott *PV*, vagy akár az összes (`-a` kapcsoló) „kioszthatóságát” engedélyezni illetve tiltani tudjuk. A *man* oldalon is megemlítik, hogy ez „rosszalkodó” merevlemezek esetén hasznos, hiszen nem tudjuk a letiltott *PV*-ket újra felhasználni. Akinek nehezére esik ezeket a programokat megjegyezni, annak ajánlom az `lvm` parancsot, hiszen az összes fent említett programocska az `lvm` szimbolikus láncait. Csak gépeljük be hogy `lvm`, és már láthatjuk is az elérhető funkciók listáját.

Végszó

Manapság nagy divat lett az *EVMS* (*Enterprise Volume Management System*), mely igazából a *LVM* leváltására készült, de azért nem szabad lemondani az *LVM*-ről sem (amúgy is az *EVMS* kezeli az *LVM* köteteket is, így később is át lehet állni az új megoldásra). Ha egyszer megszoktuk a *PV, LV, VG* elnevezéseket, akkor már biztonságosan és könnyedén leszünk képesek ezt a technológiát használni. Az *LVM* egyik súlyos problémája, hogy hihetetlenül alul dokumentált projekt, ráadásul a bináris segédprogramok üzenetei is legtöbbször semmit mondóak (ezért írtam le a legtöbb program kimenetét, sokszor nehéz észrevenni, hogy valami nem stimmel). Remélem, használható anyagot sikerült összehoznom, és még jobban remélem, hogy sok *Linux* szerető ember hasznára válnak az itt leírtak.

Gráma Tibor
(tibor.grama@hoya.lmh.hu)

1997 óta „Linuxozik”, UHU hívő. Szabadidejében gyermekeivel és vizsla kutyáival játszik, ha éppen nem kertészkedik vagy horgászik.

KAPCSOLÓDÓ CÍMEK

- http://cs.ubc.ca/~anirbans/cs508/508-report.pdf
- http://www.sulinet.hu/tart/cikk/Rda/0/25831/1
- http://tldp.org/HOWTO/LVM-HOWTO/
- http://librenix.com/?page=LVM