Konfiguracja i objaśnienie RAID oraz LVM

W tym artykule postaram się wam przybliżyć **RAID** (**R**edundant **A**rray of **I**ndependent **D**rives) oraz **LVM** (**L**ogical **V**olume **M**anagment). Te dwie technologie mogą w znaczący sposób poprawić zarówno wydajność jak i elastyczność dostępu do pamięci dyskowej.
Ogólnie mówiąc RAID polega na połączeniu dwóch lub więcej ilości dysków w tzw. macierz, w celu osiągnięcia dodatkowych możliwości i większego bezpieczeństwa danych niż przy użyciu pojedynczych dysków.
Tip:
W artykule tym będę omawiał jedynie programowy RAID, który jest o wiele wolniejszy od sprzętowego. Jednak dla tych z was, którzy nie posiadają kontrolera RAID lub płyty głównej z wbudowaną obsługą RAID może to być prosty i tani sposób na przetestowanie możliwości tej technologii.
Z kolei LVM jest metodą dzielenia dysków na partycje, która jest o wiele bardziej elastyczna niż standardowy podział na partycję. LVM pozwala na łączenie ze sobą fizycznych partycji w tzw. wirtualne partycje oraz umożliwia zmianę ich wielkości bez konieczności restartowania komputera.

RAID wprowadzenie

RAID jest ogólną nazwą wielu rodzajów (poziomów) macierzy dyskowych. Poniżej omówię najpopularniejsze z nich.

LINEAR RAID (APPEND)

Tworzy jedną durzą wirtualną partycję z dwóch lub więcej fizycznych partycji. Dane nie są przeplatane pomiędzy dyskami (interleaved) ani duplikowane. Ten poziom nie zapewnia typowych korzyści RAID poza możliwością stworzenia partycji większej od największego posiadanego dysku / partycji.

RAID-0 (STRIPPING)

Poziom 0 jest bardzo podobny do LINEAR RAID z tym, że dane zapisywane na dysku są przeplatane na wszystkie fizyczne partycje wchodzące w skład macierzy (sposób zapisu danych przedstawiony jest na rysunku). Ponieważ dane zapisywane są jednocześnie na dwóch lub większej ilości dysków to prędkość zapisu teoretycznie wzrasta N razy (gdzie N to ilość dysków wchodzących w skład macierzy).

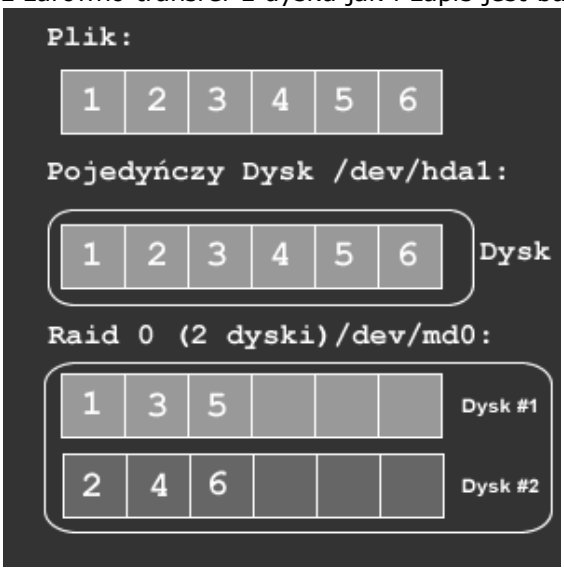
Zalety:

- wszystkie dyski widziane są jako jeden wielki dysk o powierzchni będącej sumą powierzchni dysków wchodzących w skład macierzy.
- przyspieszenie zapisu i odczytu danych w porównaniu z pojedynczym dyskiem.

Wady:

- brak odporności na awarię dysków. Wystarczy, że jeden dysk ulegnie awarii by stracić dane gromadzone na wszystkich dyskach.

RAID-0 bardzo dobrze sprawdza się jako miejsce do przechowywania dużych plików multimedialnych, ponieważ zarówno transfer z dysku jak i zapis jest bardzo szybki.



RAID-1 (MIRRORING)

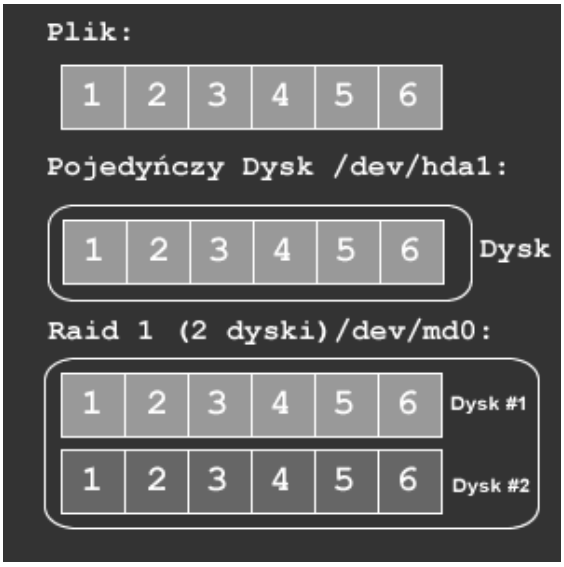
Ten poziom RAID tworzy dokładną kopie danych na jednym lub większej ilości dodatkowych dysków (rys.). Poprawia to znacząco niezawodność, ponieważ gdy jeden z dysków ulegnie uszkodzeniu mamy kompletną kopie danych na pozostałych. Jeśli w skład macierzy wchodzi N dysków to awaria N-1 dysków nie powoduje utraty danych.
Jeśli korzystamy z implementacji jądra do stworzenia tego poziomu RAID ponosimy również straty na czasie zapisu, ponieważ system musi zapisać dane przynajmniej dwukrotnie.

Zalety:

- Zwiększona niezawodność. W macierzy składającej się z 3 dysków jeśli awarii ulegną dwa dyski to cała macierz nadal działa.
- W pewnych okolicznościach szybkość odczytu z RAID-1 może być porównywalna z odczytam RAID-0 (round-robin).

Wady:

- Zmniejszona szybkość zapisu.
- Pojemność mniejsza niż suma pojemności dysków wchodzących w skład macierzy. Znaczy to, że jeśli mamy 3 dyski o pojemności 1GB to po połączeniu ich w RAID-1 otrzymujemy macierz o pojemności 1GB.



RAID-3

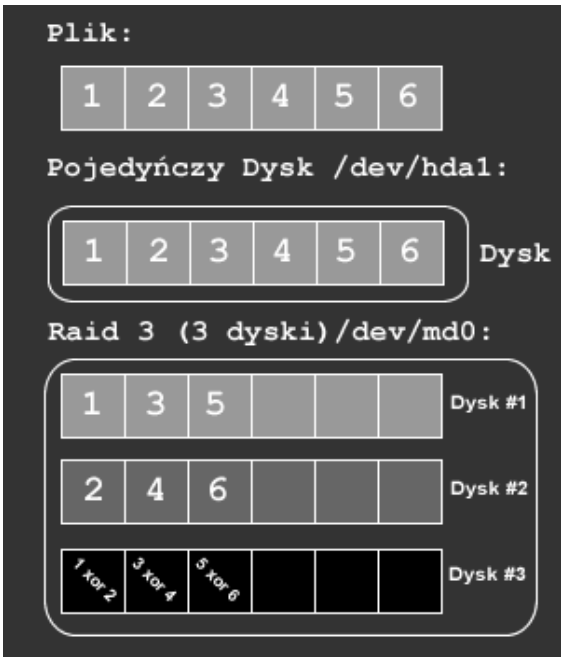
Działa na podobnej zasadzie jak RAID-0 z tym, że w macierzy istnieje dodatkowy dysk przechowujący sumy kontrolne (rys.).

Zalety:

- Odporność na awarie jednego dysku.
- Zwiększona szybkość odczytu.

Wady:

- Zmniejszona prędkość zapisu z powodu konieczności obliczania sum kontrolnych (eliminowana przy zastosowaniu sprzętowego kontrolera RAID-3).
- W przypadku awarii dysku dostęp do danych jest spowolniony z powodu obliczeń sum kontrolnych.
- Odbudowa macierzy po wymianie dysku jest operacją kosztowną obliczeniowo i powoduje spowolnienie operacji odczytu i zapisu.
- Pojedynczy, dedykowany dysk na sumy kontrolne zazwyczaj jest wąskim gardłem w wydajności całej macierzy.



RAID-4

RAID-4 jest bardzo zbliżony do RAID-3, z tą różnicą, że dane są dzielone na większe bloki (16,32, 64 lub 128 KB).

RAID-5

Ten rodzaj macierzy RAID próbuje połączyć zalety RAID-0 oraz RAID-1. RAID-5 zapisuje dane podobnie jak RAID-0 jednocześnie na wszystkich dyskach należących do macierzy ale dodaje do nich również bloki z sumami kontrolnymi, które mogą być użyte do odzyskania danych gdy nastąpi awaria jednego z dysków. Różnica między RAID-4 i RAID-5 polega na tym ze ten pierwszy zapisuje sumy kontrolne na dodatkowym dysku natomiast ten ostatni przechowuje sumy kontrolne na dyskach należących do macierzy dysków. RAID-5 podprawia transfer jak i niezawodność kosztem dodatkowego dysku.

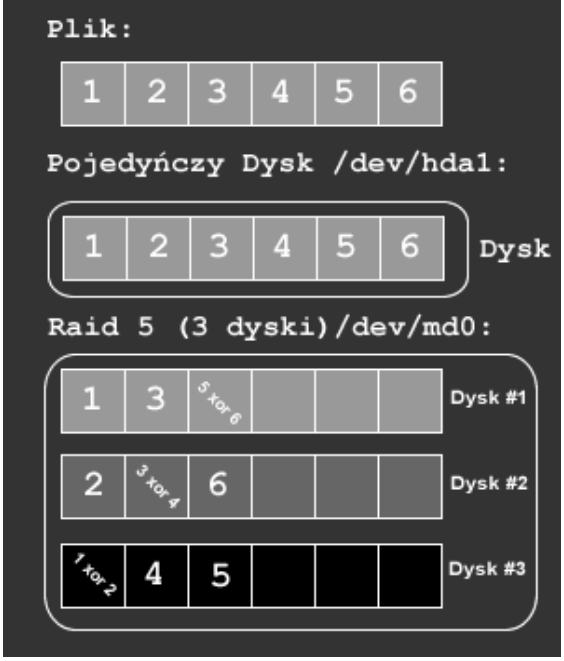
Zalety:

- Odporność na awarię 1 dysku.
- Zwiększona szybkość odczytu - porównywalna do macierzy RAID-0.

Wady:

- Zmniejszona szybkość zapisu z powodu konieczności kalkulowania sum kontrolnych (eliminowana poprzez zastosowanie sprzętowego kontrolera RAID-5).
- W przypadku awarii jednego z dysków dostęp do danych jest spowolniony z powodu konieczności obliczania sum kontrolnych.
- Odbudowa macierzy po wymianie dysku jest operacją kosztowną obliczeniowo i powoduje spowolnienie operacji odczytu i zapisu.

RAID-5 jest dość niezawodny ale awaria dwóch dysków powoduje utracenie danych. RAID-6 dodaje jeszcze więcej niezawodności kosztem dodania kolejnego dysku z sumami kontrolnymi (potrzebne jest N+2 dysków by uzyskać pojemność N dysków).



RAID-6

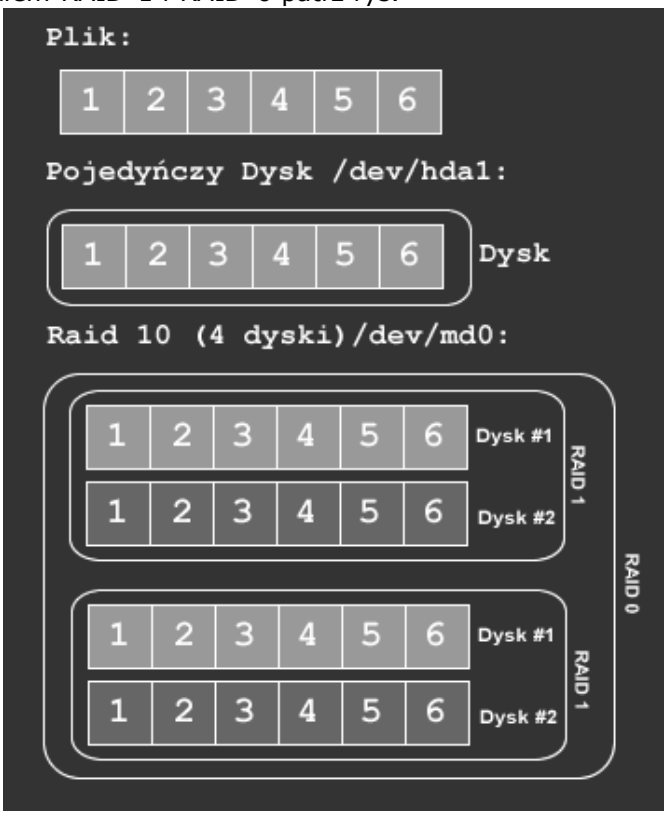
RAID-6 jest bardziej niezawodnym odpowiednikiem RAID-5. Niezawodność poprawiono dodając kolejny dysk przechowujący sumy kontrolne.

Zalety:

- Odporność na awarię wielu dysków.
- Szybkość pracy większa niż szybkość pojedynczego dysku.
- Ekstremalnie wysokie bezpieczeństwo.

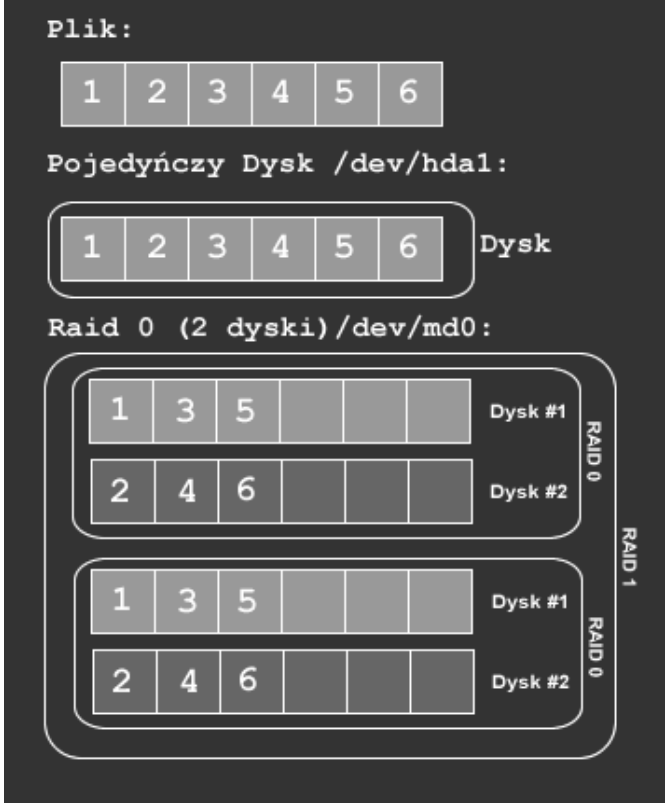
RAID-10

RAID-10 jest połączeniem RAID-1 i RAID-0 patrz rys.



RAID-01

RAID-01 jest połączeniem RAID-1 i RAID-0 patrz rys.

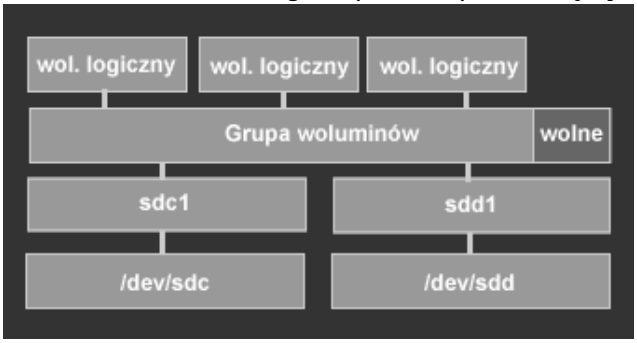


LVM wprowadzenie

LVM umożliwia nam bardziej elastyczne zarządzanie partycjami a w szczególności ich wielkością. Używając LVM jesteśmy w stanie zmieniać wielkość partycji na życzenie, bez konieczności restartowania komputera. Przy niektórych systemach plików nie musimy nawet odłączać partycji by zmienić jej wielkość.

Tip:
Obecnie ext2fs oraz ext3fs muszą być odłączone by zmienić ich rozmiar; pojemność ReiserFS może być zmieniana zarówno gdy partycja jest podłączona jak i odłączona; JFS i XFS muszą być podłączone.

Zasada działania LVM jest dość prosta do zrozumienia. Wszystkie partycje wchodzące w skład naszego systemu nazywać będziemy woluminami fizycznymi (ang. phisical volume). Na rysunku przedstawione są one jako sdc1 oraz sdd1. Jak nazwa wskazuje są to fizyczne jednostki podziału dysku. Partycje grupowane są przez LVM w tzw. grupy woluminów (ang. volume group). Grupy woluminów można porównać do wirtualnych dysków, którymi będziemy operować z poziomu LVM. Na każdym z takich wirtualnych dysków tworzyć będziemy tzw. woluminy logiczne (ang. logical volume) będące swego rodzaju wirtualnymi partycjami, które z kolei będą mogły być powiększane lub zmniejszane na życzenie. Na naszym przykładowym rysunku stworzyliśmy tylko jedną grupę woluminów ale równie dobrze mogliśmy stworzyć ich więcej.



Dodatkową zaletą LVM jest możliwość skonfigurowania go w podobny sposób do RAID-0, czyli by zapisywane dane były przeplatane pomiędzy dyskami. Poprawiając dzięki temu czas dostępu i zapisu danych jeśli do grupy woluminów należą woluminy fizyczne znajdujące się na różnych dyskach.

Wolne miejsce w grupie woluminów pozostawiłem świadomie. Przyda się ono w momencie gdy będziemy chcieli powiększyć rozmiar jednego z woluminów logicznych. Można oczywiście przydzielić całe wolne miejsce w grupie woluminów dla woluminów logicznych ale gdy będziemy chcieli powiększyć rozmiar jednego z nich to będziemy musieli zmniejszyć rozmiar innego - a jest to o wiele bardziej skomplikowane niż powiększanie woluminu.

Dobre rady

Zarówno LVM jak i RAID wymagają wsparcia jądra do poprawnego działania. Jak wiesz jądro ładowane jest z dysku podczas startu. Dlatego powinieneś się dobrze zastanowić nad umieszczaniem ważnych, z punktu widzenia startu systemu, katalogów na LVM lub RAID o wyższych poziomach niż 1. Katalogi takie jak / (root), /boot, /etc, /root, /bin, /sbin, /mnt, /dev najlepiej trzymać poza nimi. Nie znaczy to, że nie da się tego zrobić. W przypadku poziomów RAID większych niż 1 załadowanie systemu z macierzy wymaga stworzenia ramdrive i może być dość skomplikowane. Umieszczenie wspomnianych katalogów na oddzielnej partycji może znacząco ułatwić dostęp do wielu potrzebnych i przydatnych programów w razie jakiejkolwiek awarii.

Przygotowanie systemu

Jeżeli chodzi o Slackware to nie mamy zbyt dużo do przygotowania. Wszystkie pre-kompilowane jądra na Slackware CD prócz lowmem.i mają wkompilowaną obsługę RAID i LVM. Jeśli nie jesteś pewny czy twoje jądro obsługuje RAID i LVM, wystarczy, że sprawdzisz czy w katalogu /proc istnieją pliki /proc/mdstat (RAID), /proc/lvm/global (LVM). Jeśli istnieją to masz jądro obsługujące te dwie technologie.
Wkompilowane w jądro sterowniki to jedynie połowa sukcesu. Do obsługi i konfiguracji LVM oraz RAID będziemy również potrzebować programów zawartych w poniższych pakietach:
installpkg lvm-1.0.8-i486-1.tgz
installpkg mdadm-2.1-i486-1.tgz
Teraz mamy wszystko, co potrzeba by przystąpić do konfiguracji.

Własne jądro

Jeśli zamierzasz kompilować własne jądro to przy konfiguracji musisz włączyć *Multi-device support (RAID and LVM) -> RAID support* (jądra 2.4) oraz wybrać interesujący cię poziom RAID. Radzę wkompilować jego obsługę w jądro a nie jako moduł. Dodatkowo należy zaznaczyć *Logical volume manager (LVM)* support (jadra 2.4).

Przykładowy system

W celu zaprezentowania w jaki sposób skonfigurować RAID oraz LVM przygotowałem system testowy zawierający 4 dyski SCSI:
/dev/sda1 - dysk nr 0, 10GB, ReiserFS, system
/dev/sda2 - dysk nr 0, 70MB, ReiserFS, swap
/dev/sdb - dodatkowy dysk nr 1, 10GB
/dev/sdc - dodatkowy dysk nr 2, 10GB
/dev/sdd - dodatkowy dysk nr 3, 10GB
Dysk nr 0 zawiera system Slackware z jądrem 2.4.31 i obsługą SCSI - jądro scsi.s. Pozostałe dyski zostały dodane później i na razie nie zawierają żadnych partycji.
Tip:
Jeśli chodzi o RAID to nie ma znaczenia czy tworzymy macierz z całych dysków (np. /dev/sdd) czy z konkretnych partycji (np. /dev/sdd1).

Do dzieła

Każdy projekt zaczynamy od oceny sytuacji i planowania. W naszym przypadku oceną sytuacji jest to, że posiadamy cztery dyski, każdy o pojemności 10GB. Na jednym z nich mamy już zainstalowany system, pozostałe są puste. Nasz projekt będzie miał na celu stworzenie dwóch macierzy RAID a następnie umieszczenie na nich pewnych części katalogu głównego (/). Pokażę również jak skonfigurować start systemu z macierzy RAID-1. Nasz projekt podzielimy na kilka faz. Każda z faz pomyślana jest tak by nauczyć jak najwięcej zarówno o technologii RAID jaki i LVM. Przypominam, że jest to jedynie przykład a podejmowane przeze mnie decyzje o podziale na partycje, tworzeniu macierzy RAID, tworzeniu grup woluminów, oraz partycji logicznych oparte są przede wszystkim na tym by pokazać jak najwięcej możliwość tych technologii. Nie znaczy to, że przyjęty przeze mnie podział jest zły lub nieopłacalny.

Faza 1. Tworzymy RAID-1 na dyskach 2 i 3

Zaczniemy od stworzenia odpowiednich partycji:

```
# fdisk /dev/sdc
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.
```

```
The number of cylinders for this disk is set to 1305.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)


Command (m for help): n <- Nowa partycja
Command action
   e   extended
   p   primary partition (1-4)
p <- Primary partition
Partition number (1-4): 1
First cylinder (1-1305, default 1): 1 <- Lub naciskamy ENTER by wybrać 1 domyślnie
Last cylinder or +size or +sizeM or +sizeK (1-1305, default 1305): 1305 <- Lub naciskamy ENTER

Command (m for help): t <- Zmieniamy rodzaj partycji
Selected partition 1
Hex code (type L to list codes): L <- Wyświetlamy dostępne możliwości

 0  Empty           1e  Hidden W95 FAT1 80  Old Minix       be  Solaris boot
 1  FAT12           24  NEC DOS         81  Minix / old Lin bf  Solaris
 2  XENIX root      39  Plan 9          82  Linux swap      c1  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  83  Linux           c4  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 5  Extended        41  PPC PReP Boot   85  Linux extended  c7  Syrinx
 6  FAT16           42  SFS             86  NTFS volume set da  Non-FS data
 7  HPFS/NTFS       4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
 8  AIX             4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
 9  AIX bootable    4f  QNX4.x 3rd part 8e  Linux LVM       df  BootIt
 a  OS/2 Boot Manag 50  OnTrack DM      93  Amoeba          e1  DOS access
 b  W95 FAT32       51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
 c  W95 FAT32 (LBA) 52  CP/M           9f  BSD/OS          e4  SpeedStor
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a5  FreeBSD         ee  EFI GPT
10  OPUS            55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       a9  NetBSD          f4  SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys ab  Darwin boot     f2  DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fd  Linux raid auto
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fe  LANstep
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid ff  BBT
1c  Hidden W95 FAT3 75  PC/IX
Hex code (type L to list codes): fd <- Wybieramy Linux raid auto by partycja mogła być wykryta
Changed system type of partition 1 to fd (Linux raid autodetect)

Command (m for help): p <- Wyświetlamy nasze zmiany

Disk /dev/sdc: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1               1        1305    10482381   fd  Linux raid autodetect

Command (m for help): w <- Zapisujemy zmiany
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
#
```

Tak samo postępujemy w przypadku /dev/sdd. Jeśli wszystko przebiegnie pomyślnie to mamy

przygotowane partycje by połączyć je w macierz RAID. Naszą pierwszą macierzą będzie RAID-1. Tworzymy ją z myslą o umieszczeniu na niej ważnych danych, które muszą przetrwać awarie jednego z dysków. Dla nas będą to /home, /var, /usr. Jest wiele szkół jakie katalogi powinno się umieszczać na macierzy RAID. Wiele z nich jest sprzecznych. Moim zdaniem najlepiej wypracować sobie własne podejście do tego problemu.
Jesteśmy gotowi do skonfigurowania macierzy RAID-1. Wykorzystujemy do tego polecenie mdadm, które ma postać:
mdadm [tryb] <nazwa_macierzy_raid> [opcje] <partycje_składowe>
Wydajemy polecenie:
mdadm --create /dev/md1 --level=raid1 --raid-devices=2 /dev/sdc1 /dev/sdd1
mdadm: array /dev/md1 started. <- Taki komunikat znaczy że wszustko jest OK.
#
Znaczenie przełączników:
--create /dev/md1
informujemy że będziemy tworzyć macierz RAID o nazwie /dev/md1.
--level=raid1
bedzie to macierz RAID-1
--raid-devices=2
ilość aktywnych urządzeń (dysków/partycji) wchodzących w skład macierzy. Możemy dodać do macierzy dyski zapasowe (spare), które będą automatycznie dołączane do macierzy w przypadku awarii jednego z "głównych" dysków (partrz manual do mdadm: --spare-devices).
Jeśli tworzenie macierzy się powiodło to kolejnym krokiem będzie stworzenie pliku konfiguracyjnego /etc/mdadm.conf wykorzystywanego przez sterownik md w jądrze do aktywowania macierzy. Jeśli nie stworzymy tego pliku to md będzie skanował partycje zawarte w pliku /proc/partitions by stwierdzić, które z nich wchodzą w skład RAID. Jednym słowem nie jest on potrzebny, ale warto go mieć. Plik tworzymy przy pomocy polecenia:
mdadm --detail --scan >> /etc/mdadm.conf
następnie dodajemy ręcznie linię:
DEVICE /dev/sdc /dev/sdd
lub krócej:
DEVICE /dev/sd[cd]
Informuje ona md jakie urządzenia wchodzą w skład macierzy. Poprawny plik w naszym przypadku powinien wyglądać mniej więcej tak:
DEVICE /dev/sd[cd]
ARRAY /dev/md1 level=raid1 num-devices=2 UUID=a7990811:9d4d4f50:0524d61e:cc7ddae1
Jedyną różnicą w waszym pliku powinien być numer UUID (**U**niversally **U**nique **ID**entifier), który jest 128-bitowym numerem mającym gwarantowaną unikalność w naszym oraz innych systemach. Jest on generowany losowo na podstawie posiadanego w systemie sprzętu oraz wartości zegara systemowego. UUID jest wykorzystywany przez wiele programów do unikalnego oznaczania urządzeń. Możemy sami wygenerować taki numer używając polecenia uuidgen.

Status macierzy

Jeśli chodzi o sprawdzanie statusu macierzy to mamy kilka możliwości. Pierwszą z nich jest wyświetlenie pliku /proc/mdstat.

```
# cat /proc/mdstat
Personalities : [linear] [raid0] [raid1] [raid5]
read_ahead 1024 sectors
md1 : active raid1 sdd1[1] sdc1[0]
      10482304 blocks [2/2] [UU]

unused devices:
```

Jak widać plik zawiera wiele informacji na temat działających macierzy RAID. Zacznijmy od trzeciej linii, która informuje nas, że macierz /dev/md1 jest macierzą RAID-1, jest aktywna i składa się z dwóch partycji /dev/sdd1 oraz /dev/sdc1. Liczby w nawiasach kwadratowych po nazwach partycji informują jaka jest rola lub funkcja danej partycji w macierzy. Kolejna linijka informuje ile jest bloków w macierzy. Następny jest nawias kwadratowy w postaci [#/#] - informuje nas, że do macierzy należą 2 dyski z czego 2 są dostępne. Nawias kwadratowy zaraz za nim informuje, że oba dyski są **U**p - czyli działają. W momencie gdy jeden z dysków jest odłączony (ang. missing) w miejscu U zobaczymy "_". Jeśli natomiast mamy dysk, który się

zepsuł lub nie odpowiada z jakiegoś powodu to po nawiasie kwadratowym mówiącym o roli danego dysku w macierzy zobaczymy "**(F)**" od ang. failed.
Status macierzy możemy uzyskać również za pomocą komendy mdadm.

```
# mdadm -Q /dev/md1
/dev/md1: 9.100GiB raid1 2 devices, 0 spares. Use mdadm --detail for more detail.
# mdadm -D /dev/md1
/dev/md1:
        Version : 00.90.00
  Creation Time : Tue Apr  4 00:26:44 2006
     Raid Level : raid1
     Array Size : 10482304 (10.00 GiB 10.73 GB)
    Device Size : 10482304 (10.00 GiB 10.73 GB)
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 1
    Persistence : Superblock is persistent

    Update Time : Tue Apr  4 00:26:44 2006
          State : active
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           UUID : a487e866:701d5e6b:2e3a44a3:aabc9fb1
         Events : 0.2

    Number   Major   Minor   RaidDevice State
       0       8       33        0      active sync   /dev/sdc1
       1       8       49        1      active sync   /dev/sdd1
```

Oprócz dwóch pokazanych wyżej opcji istnieje również opcja -E ale przyjmuje jako argument nie nazwę macierzy tylko nazwę partycji składowych - np. /dev/sdd1.

Podsumowanie

Jeśli zrobiłeś wszystko tak jak opisałem powinieneś mieć działającą macierz RAID-1 o nazwie /dev/md1. Jeśli nie planowalibyśmy użyć LVM to mógłbyś w tym momencie sformatować ją przy pomocy mkfs i zacząć używać jak każdy inny dysk.

Konfigurujemy LVM

Jak pamiętasz LVM składa się z trzech "warstw" - woluminów fizycznych, grup woluminów oraz woluminów logicznych. Każdy z tych składników musi być odpowiednio przygotowany do pracy z LVM. Naszym woluminem fizycznym będzie macierz /dev/md1. Aby ją przygotować wydajemy komendy:

```
# vgscan -v
vgscan -- removing "/etc/lvmtab" and "/etc/lvmtab.d"
vgscan -- creating empty "/etc/lvmtab" and "/etc/lvmtab.d"
vgscan -- reading all physical volumes (this may take a while...)
vgscan -- "/etc/lvmtab" and "/etc/lvmtab.d" successfully created
vgscan -- WARNING: This program does not do a VGDA backup of your volume group

# pvcreate /dev/md1
pvcreate -- physical volume "/dev/md1" successfully created
```

Pierwszą z nich wydajemy tylko po to by stworzyć plik /etc/lvmtab oraz katalog /etc/lvmtab.d. Gdybyśmy jej nie wydali polecenie pvcreate zwróciłoby błąd. Polecenie pvcreate można porównać do polecenia mkfs. Przygotowuje ono partycję do późniejszego wykorzystania przez LVM. W źródłach opisujących LVM można przeczytać, że każda partycja wchodząca w skład woluminu fizycznego powinna być typu 0x8E (Linux LVM). W przypadku partycji RAID jest to bezsensowne. Tym bardziej, że partycjom wchodzącym w skład RAID ustawiliśmy już rodzaj (ID) na 0xFD.
Ostatnia linia wyniuku działania polecenia vgscan zawiera ostrzerzenie, że ten program nie robi

kopii zapasowej VGDA (**V**olume **G**roup **D**escriptor **A**rea). VGDA jest czymś podobnym do tablicy partycji. VGDA umieszczone jest na początku każdego woluminu fizycznego wchodzącego w skład LVM i zawiera dane dotyczące jego konfiguracji. Podczas startu systemu woluminy logiczne oraz grupy woluminów zostają aktywowane a VGDA ładowany jest do pamięci. VGDA pomaga w zidentyfikowaniu gdzie na woluminach fizycznych znajdują się woluminy logiczne oraz w jaki sposób mapowane są bloki logiczne (LE) na bloki fizyczne (PE).

Grupy woluminów

Kolejnym krokiem jest stworzenie grupy woluminów przy pomocy komendy vgcreate:

```
# vgcreate -s 32M vg00 /dev/md1

  vgcreate -- INFO: maximum logical volume size is 2 Terabyte
  vgcreate -- doing automatic backup of volume group "vg00"
  vgcreate -- volume group "vg00" successfully created and activated
```

Ostatnim parametrem powyższej komendy jest nasza macierz, przed ostatnim nazwa jaką będzie miała nasza grupa woluminów. Pierwszy parametr (-s 32M) to wielkość bloku fizycznego (ang. Phisical Extent Size w skrócie PE), którego znaczenia jeszcze nie wyjaśniłem. Każdy wolumin fizyczny wchodzący w skład grupy woluminów zostanie podzielony na bloki o podanym przez nas rozmiarze (PE). Następnie każdy stworzony w tej grupie woluminów wolumin logiczny zostanie podzielony na takie same bloki. Wielkość bloku fizycznego musi być potęgą liczby 2 i może być z przedziału 8KB do 16GB. Weźmy pod uwagę taki przykład:
Stworzyliśmy grupę woluminów o nazwie vg11 i ustawiliśmy PE równe 4MB. Do grupy należą dwie partycje /dev/hda1 i /dev/hdb1. Nazwijmy je kolejno PV1 i PV2. Każda z tych partycji podzielona zostanie na 4MB bloki ponieważ nasza grupa woluminów stworzona została z PE = 4MB. Przyjmijmy, że PV1 zawiera 50 PE a PV2 120 PE. Jeśli będziemy teraz chcieli stworzyć wolumin logiczny w grupie woluminów vg11 będzie on mógł mieć rozmiar od 1 PE do 170 PE czyli w naszym przypadku od 4MB do 680MB. Bloki przyporządkowane do woluminów logicznych nazywane są blokami logicznymi (ang. Logical Extent w skrócie *LE*). W czasie tworzenia woluminu logicznego definiowane jest mapowanie między PE a LE. Przykładowo LE nr 1 może zostać przypisany do PE nr 120 na PV1.
Kolejnym krokiem po stworzeniu grupy woluminów będzie podzielenie jej na woluminy logiczne. Nasza grupa woluminów składa się z 380 bloków fizycznych po 32MB każdy. Daje nam to w sumie 9.94GB do podziału między woluminy logiczne. Możemy to bardzo łatwo sprawdzić wydając komendę:

```
# vgdisplay
--- Volume group ---
VG Name               vg00
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                0
Open LV               0
MAX LV Size           2 TB
Max PV                256
Cur PV                1
Act PV                1
VG Size               9.94 GB  <-
PE Size               32 MB  <-
Total PE              318
Alloc PE / Size       0 / 0
Free  PE / Size       318 / 9.94 GB  <-
VG UUID               9k4l4i-N02z-MUo7-fXe4-CQ9q-GANa-dV4Wc4
```

Wolne miejsce podzielimy następująco: partycja przeznaczona na /var 300MB, partycja /usr 3GB, partycja /home 2GB. Resztę miejsca pozostawimy nieprzydzieloną by móc rozszerzać wspomniane dyski logiczne. Jak już wspomniałem wcześniej o wiele łatwiej jest rozszerzać partycje niż zmniejszać. Rozdzielając całe dostępne miejsce między woluminy logiczne zmusiłoby nas później do zmniejszania jednej partycji by powiększyć drugą.

Woluminy logiczne

Woluminy logiczne tworzymy przy pomocy polecenia lvcreate:

```
# lvcreate -L300M -n varlv vg00
lvcreate -- rounding size up to physical extent boundary
lvcreate -- doing automatic backup of "vg00"
lvcreate -- logical volume "/dev/vg00/varlv" successfully created
```

W tym przypadku stworzyliśmy wolumin logiczny o nazwie varlv (-n varlv) i pojemności 300MB (-L300M) w grupie woluminów vg00. Pojemność woluminów logicznych możemy podawać w kilobajtach (K), megabajtach (M), gigabajtach (G), terabajtach (T). Ostatnia linia powurzszego polecenia informuje nas, że wolumin logiczny został stworzony i możemy się do niego odwoływać za pomocą /dev/vg00/varlv.
Ciekawą opcją polecenia lvcreate, nie użytą powyżej, jest -i (--stripes). Informuje ona LVM by stworzyć wolumin logiczny przeplatany na 2 lub więcej dysków. Pozwala to stworzyć coś na wzór RAID-0. Wraz z opcją -i możesz również użyć opcji -I by sprecyzować wielkość bloku do przeplatania. Rozmiar tego bloku musi być potęgą 2 między 2 a 512 (np. 64, 256).
W podobny sposób tworzymy woluminy logiczne usrlv oraz homelv:

```
# lvcreate -L300M -n varlv vg00
# lvcreate -L3G -n usrlv vg00
```

Gdy mamy już stworzone woluminy logiczne możemy je traktować jak noramlne partycje. Pierwszym krokiem będzie stworzenie na nich systemu plików.

```
# mkfs.reiserfs /dev/vg00/varlv
mkfs.reiserfs 3.6.19 (2003 www.namesys.com)

A pair of credits:
Continuing core development of ReiserFS is  mostly paid for by Hans Reiser from
money made selling licenses  in addition to the GPL to companies who don't want
it known that they use ReiserFS  as a foundation for their proprietary product.
And my lawyer asked 'People pay you money for this?'. Yup. Life is good. If you
buy ReiserFS, you can focus on your value add rather than reinventing an entire
FS.

Alexander  Lyamin  keeps our hardware  running,  and was very  generous  to our
project in many little ways.


Guessing about desired format.. Kernel 2.4.31 is running.
Format 3.6 with standard journal
Count of blocks on the device: 81920
Number of blocks consumed by mkreiserfs formatting process: 8214
Blocksize: 4096
Hash function used to sort names: "r5"
Journal Size 8193 blocks (first block 18)
Journal Max transaction length 1024
inode generation number: 0
UUID: a577bf01-7f74-46ef-a94f-d39cc86daef8
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
        ALL DATA WILL BE LOST ON '/dev/vg00/varlv'!
Continue (y/n):y
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..ok

Tell your friends to use a kernel based on 2.4.18 or later, and especially not a
kernel based on 2.4.9, when you use reiserFS. Have fun.

ReiserFS is successfully created on /dev/vg00/varlv.
```

W ten sam sposób tworzymy systemy plików na woluminach homelv i usrlv.

Przenosimy katalogi

W tym momencie będę was musiał troche zmartwić. Aby przenieść katalogi /var, /home oraz /usr będziemy musięli zrestartować naszą maszyne - a szkoda miałęm taki łądny uptime. Pytacie dlaczego? Otóż wyżej wymienione katalogi są dość intensywnie urzywane zaróweno przez system jaki i użytkowników i nie jesteśmy w stanie przenierść ich bez utraty danych. Wyobrażmy sobie taką sytuację: Jesteśmy przy końcu kopiowania katalogu /var a między

czasie jeden z procesów systemowych zapisuje dane do już skopiowanego pliku w "oryginalnym" katalogu /var. Tracimy w tym momencie ważne informacje. To samo tyczy się pozostałych katalogów.
Najlepszym wyjściem z tej sytuacji jest zatrzymanie systemu oraz uruchomienie go z Slackware CD. Następnie podpięcie naszego katlogu głównego oraz wcześniej przygotowanych woluminów logicznych i przekopiowanie interesujących nas danych. Po czym skasowanie niepotrzebnych plików z katalogów /var, /home oraz /usr i odpowiednie zmoidyfikowanie pliku /etc/fstab.
Instrukcję krok po kroku jak to zroić przedstawiam poniżej.
Startujemy z CD-ROM dokładnie w ten sam sposób jak przy instalacji (niezapomij o wybraniu odpowiedniego do twojego systemu obrazu jądra) z tą rużnicą ze po wybraniu klawiatóry oraz logowaniu jako root wydajemy następujące komandy:

```
# mkdir /mnt/root
# mkdir /mnt/homelv
# mkdir /mnt/varlv
# mkdir /mnt/uselv
# mount /dev/sda1 /mnt/root
# cd /mnt/root
# chroot .
#
```
