Администрирование Linux. Часть2.

Пособие для дистанционных курсов.
http://linuxnavigator.ru.

01.07.2009

Оглавление

РАБОТА С НАКОПИТЕЛЯМИ.

Во время установки мы разбили на разделы только один жесткий диск. Второй диск был оставлен нетронутым, планируя в дальнейшем использовать его в программном RAID. Конечно, создать RAID можно было и на этапе установки, но мне очень хотелось показать вам инструменты для работы с накопителями. В том числе и для работы с программным RAID.

Сразу хочу предупредить, что создание RAID немного замедлит работу дисковой подсистемы, из-за того, что виртуальная машина, в которой запущен ваш сервер, работает на одном физическом диске. На реальной машине все будет работать нормально.

Прежде чем мы начнем создание RAID, рассмотрим некоторые инструменты, позволяющие работать с накопителями.

РАЗБИЕНИЕ ДИСКА НА РАЗДЕЛЫ.

Для разбиения диска на разделы в Linux можно использовать несколько различных программ, но мы рассмотрим только программу fdisk, которая присутствует практически во всех дистрибутивах.

Для начала посмотрим, какие накопители у нас установлены, и какие разделы на них созданы. Для этого достаточно запустить программу fdisk с параметром –l.

```
# fdisk -l


Диск /dev/sda: 15.0 ГБ, 15032385536 байт

255 heads, 63 sectors/track, 1827 cylinders

Единицы = цилиндры по 16065 * 512 = 8225280 байт


Устр-во Загр     Начало       Конец       Блоки   Id  Система

/dev/sda1   *           1          13      104391   83  Linux

/dev/sda2              14         650     5116702+  83  Linux

/dev/sda3             651         781     1052257+  82  Linux своп / Solaris

/dev/sda4             782        1827     8401995    5  Расширенный

/dev/sda5             782         912     1052226   83  Linux

/dev/sda6             913        1827     7349706   83  Linux


Диск /dev/sdb: 15.0 ГБ, 15032385536 байт

255 heads, 63 sectors/track, 1827 cylinders

Единицы = цилиндры по 16065 * 512 = 8225280 байт
```

Как видно из вывода программы, диск sda разбит на разделы, а диск sdb разделов не имеет.

То, что мы будем делать сейчас, не имеет никакого отношения к созданию программного RAID. Для начала мы просто посмотрим, как работать с накопителями в Linux. Для начала мы создадим несколько разделов на диске sdb, отформатируем их (создадим там файловую систему) и научимся ими пользоваться.

Для разбиения диска sdb на разделы необходимо запустить программу fdisk, указав в качестве параметра файл устройства диска. Нас ожидает один неприятный момент, связанный с русификацией системы — при выводе на экран, будут разъезжаться колонки. Поэтому мы запустим программу с английским интерфейсом. Для этого, перед именем программы указываются переменные локализации. Командная строка будет выглядеть следующим образом:

```
LANG=POSIX fdisk /dev/sdb
```

Запустим программу.

```
# LANG=POSIX fdisk /dev/sdb

The number of cylinders for this disk is set to 1827.
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):
```

У программы есть своя собственная командная строка — *Command (m for help):*

Для получения краткого списка команд воспользуйтесь командой *m*. Введите *m* и нажмите Enter.

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Создадим один первичный раздел, один расширенный и в нем один логический раздел.

Для создания раздела используем команду *n*. Программа попросит вас указать, какой тип раздела вы собираетесь создавать, первичный или расширенный. Для создания первичного раздела надо нажать *p* и *Enter*.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
```

Затем вы должны указать номер создаваемого раздела, введите 1 и нажмите *Enter*.

```
Partition number (1-4): 1
```

Теперь вам необходимо указать границы раздела. Границы определяются номером начального и конечного цилиндров. На приглашение ввести первый цилиндр, просто нажмите *Enter*. Программа использует значение по умолчанию. В нашем случае — это первый цилиндр.

```
First cylinder (1-1827, default 1):
Using default value 1
```

Затем введите последний цилиндр. Для того, что бы не мучатся и не вычислять его номер, можно просто указать символ плюс и размер создаваемого раздела. Создадим раздел размером 200M.

```
Last cylinder or +size or +sizeM or +sizeK (1-1827, default 1827): +200M
```

И вы снова получаете приглашение командной строки программы.

Посмотрим список разделов при помощи команды p.

```
Command (m for help): p

Disk /dev/sdb: 15.0 GB, 15032385536 bytes
255 heads, 63 sectors/track, 1827 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          25      200781   83  Linux

Command (m for help):
```

Был создан один раздел — */dev/sdb1*. Его размер показан в столбце *Blocks*. Один блок равен одному килобайту.

Теперь создадим расширенный раздел, для этого воспользуемся командой *n*.

На вопрос о типа создаваемого раздела, нажмем *e*.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
```

Номер раздела будет 2.

```
Partition number (1-4): 2
```

Первый цилиндр по умолчанию — *Enter*.

```
First cylinder (26-1827, default 26):
Using default value 26
```

Последний цилиндр, то же по умолчанию *Enter*. Таким образом, мы выделим под раздел все оставшееся дисковое пространство.

```
Last cylinder or +size or +sizeM or +sizeK (26-1827, default 1827):
Using default value 1827
```

При помощи команды *p* посмотрим, что у нас получилось.

```
Command (m for help): p

Disk /dev/sdb: 15.0 GB, 15032385536 bytes
```

```
255 heads, 63 sectors/track, 1827 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          25      200781   83  Linux
/dev/sdb2              26        1827    14474565    5  Extended
```

Создадим логический раздел размером 1Gb. Воспользуемся командой *n*. Обратите внимание на то, что вы не сможете создавать расширенный разделы. Мы можем делать только логические разделы при помощи команды *l*. Нажмите *l*.

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
```

Первый цилиндр по умолчанию *Enter*.

```
First cylinder (26-1827, default 26):
Using default value 26
```

Последний цилиндр *+1024M* или *+1G*.

```
Last cylinder or +size or +sizeM or +sizeK (26-1827, default 1827): +1024M
```

Командой *p* смотрим, что у нас получилось.

```
Command (m for help): p

Disk /dev/sdb: 15.0 GB, 15032385536 bytes
255 heads, 63 sectors/track, 1827 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          25      200781   83  Linux
/dev/sdb2              26        1827    14474565    5  Extended
/dev/sdb5              26         150     1004031   83  Linux
```

Обратите внимание на то, что в Linux первый логический раздел всегда имеет номер 5.

Предположим, что я собираюсь использовать раздел *sdb1* под swap пространство. В этом случае я должен изменить тип раздела. В столбце *Id* указан тип раздела. Для всех разделов, кроме расширенного, по умолчанию устанавливается тип *83 — Linux*. Эти разделы предназначены для создания в них файловых систем (форматирования).

Давайте посмотрим какие тпы разделов можно установить. Для этого воспользуется командой *l*.

```
Command (m for help): l

 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 / So 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
```

Как видите их достаточно много. Но реально мы будем пользоваться только четырьмя из них:

- 82 — предназначены для создания swap.
- 83 — предназначены для создания файловых систем.
- 8e — предназначены для использования в LVM (Logical Volume Manager).
- fd — предназначены для использования в программном RAID.

Запомните тип fd, мы им позднее воспользуемся. А сейчас при помощи команды *t* изменим тип раздела *sdb1* на *82*.

Введите команду t. Затем укажите номер раздела, которому вы собираетесь менять тип. Затем введите номер 82.

```
Command (m for help): t
Partition number (1-5): 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)
```

При помощи команды p проверьте, что у вас получилось.

```
Command (m for help): p

Disk /dev/sdb: 15.0 GB, 15032385536 bytes
255 heads, 63 sectors/track, 1827 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          25      200781   82  Linux swap / Solaris
/dev/sdb2              26        1827    14474565    5  Extended
/dev/sdb5              26         150     1004031   83  Linux
```

Как вы видите, раздел сменил свой тип на 82.

Если при разбиении диска вы допустили ошибку, можно выйти без сохранения, нажав комбинацию *Ctrl+C* или воспользовавшись командой *q* (выход без сохранения).

Теперь необходимо записать все изменения, которые мы сделали. Пока все это храниться в памяти программы. Воспользуемся командой *w* (записать и выйти).

```
Command (m for help): w
The partition table has been altered!

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

Мы сохранили все изменения и вышли из программы.

SWAP.

В предыдущем разделе мы создали раздел */dev/sdb1* предназначенный для *swap*. Сейчас мы научимся создавать swap файлы и разделы, подключать их к текущему swap пространству и отключать.

Посмотрим, какой размер swap у нас сейчас. Используем программу *free*.

```
# free
             total       used       free     shared    buffers     cached
Mem:        515492     507236       8256          0      20372     431540
-/+ buffers/cache:      55324     460168
Swap:      1052248          0    1052248
```

Как видно из вывода программы, текущий размер swap пространства 1052248 байта, используется ноль байт.

Для того, что бы использовать новый swap раздел, его необходимо отформатировать. Это делается при помощи программы *mkswap*.

```
# mkswap /dev/sdb1
Устанавливается пространство для свопинга версии 1, размер = 205594 кБ
#
```

Для подключения раздела к текущему пространству воспользуемся программой swapon.

```
# swapon /dev/sdb1
# free
             total       used       free     shared    buffers     cached
Mem:        515492     502960      12532          0     214448     237864
-/+ buffers/cache:      50648     464844
Swap:      1253020          0    1253020
#
```

Как видите — размер swap пространства увеличился.

Кроме разделов под swap можно использовать обыкновенные файлы. Сначала создадим такой файл при помощи программы dd.

```
# dd if=/dev/zero of=swap bs=1024 count=100
100+0 записей считано
100+0 записей написано
 скопировано 102400 байт (102 kB), 0,00109 секунд, 93,9 MB/s
# ls -l swap
```

```
-rw-r--r-- 1 root root 102400 Фев 25 13:06 swap
# mkswap swap
Устанавливается пространство для свопинга версии 1, размер = 98 кБ
# swapon swap
# free
             total       used       free     shared    buffers     cached
Mem:        515492     503632      11860          0     214472     238128
-/+ buffers/cache:      51032     464460
Swap:      1253112          0    1253112
#
```

Для того, что бы посмотреть, какие устройства и файлы используется в swap, смотрите содержимое файла */proc/swaps*.

```
# cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/sda3                               partition       1052248 0       -1
/dev/sdb1                               partition       200772  0       -2
/root/swap                              file            92      0       -3
#
```

Для отключения swap пространства используется программа swapoff.

```
# swapoff swap
# swapoff /dev/sdb1
# cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/sda3                               partition       1052248 0       -1
#
```

СОЗДАНИЕ ФАЙЛОВОЙ СИСТЕМЫ.

После создания разделов, в них необходимо создать файловую систему. В Linux поддерживается огромное количество файловых систем. Но в вашем дистрибутиве далеко не все из них могут быть включены. Для того, что бы узнать, какие файловые системы поддерживаются вашим ядром, посмотрите содержимое файла */proc/filesystems*.

```
# cat /proc/filesystems
nodev   sysfs
nodev   rootfs
nodev   bdev
nodev   proc
nodev   cpuset
nodev   binfmt_misc
nodev   debugfs
nodev   securityfs
nodev   sockfs
nodev   usbfs
nodev   pipefs
nodev   futexfs
nodev   tmpfs
```

```
nodev   inotifyfs
nodev   eventpollfs
nodev   devpts
        ext2
nodev   ramfs
nodev   hugetlbfs
        iso9660
nodev   mqueue
        ext3
nodev   rpc_pipefs
nodev   autofs
#
```

Файловые системы помеченные как *nodev* — это файловые систем не требующие физического устройства, в основном виртуальные файловые системы. Из файловых систем, которые можно создавать в разделах, в дистрибутиве CentOS на данный момент можно использовать только *ext2* и *ext3*.

В принципе в дистрибутив можно добавить и другие файловый системы, но для этого потребуется перкомпиляция ядра и установка дополнительного программного обеспечения.

Для создания файловой системы используется программа mkfs.

Если в командной строке набрать mkfs и два раза нажать на табуляцию — будет показано возможное продолжение.

```
# mkfs
mkfs          mkfs.cramfs  mkfs.ext2    mkfs.ext3    mkfs.msdos   mkfs.vfat
# mkfs
```

В Linux программа *mkfs* — это оболочка, вызывающая конкретные программы для каждой файловой системы. При помощи параметра *–t* можно указать тип создаваемой файловой системы.

```
# mkfs -t ext2 /dev/sdb5
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
125696 inodes, 251007 blocks
12550 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=260046848
8 block groups
32768 blocks per group, 32768 fragments per group
15712 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 36 mounts or
```

```
180 days, whichever comes first.  Use tune2fs -c or -i to override.
#
```

Обратите внимание на некоторые поля, которые выводила программа при создании файловой системы.

- *5.00% reserved for the super user* — 5% дискового пространства (значение по умолчанию) резервируется за суперпользователем. Это пространство необходимо для того, что бы файлы в файловой системе были не фрагментированы.
- *Superblock backups stored on blocks* — При создании файловой системы, создается так называемый суперблок. В нем хранятся основные параметры ФС и находится корень системы. Все остальное дисковое пространство будет использоваться по мере необходимости. Копия суперблока равномерно сохраняется по файловой системе. Это необходимо для восстановления ФС в случае потери основного суперблока.
- Последние строки. Эта файловая система будет автоматически проверяться через каждые 36 подключений (число выбирается случайным образом для каждой файловой системы) или через 180 дней (полгода), в зависимости от того, какое событие наступит раньше. Эти параметры необходимы для того, что бы при перезагрузке системы автоматически включалась проверка файловых систем.

Создание файловой системы произошло очень быстро. Это объясняется особенностью структуры ФС в Linux. По существу были созданы только суперблок и его копии. Остальное дисковой пространство будет использоваться по мере необходимости.

Так же не было создания кластеров, как это делается в Windows. В этом нет необходимости, ведь жесткий диск уже отформатирован на заводе, и в дальнейшем драйвер ФС будет пользоваться этими блоками.

Что бы проверить раздел на наличие ошибок, можно воспользоваться программой *badblocks*. Но при создании файловых систем типа ext2 и ext3 можно вызвать эту программу автоматически из *mkfs*. Для этого используется параметр –c. Если параметр указать один раз — будет осуществлена проверка в режиме только для чтения. Если параметр указать два раза — в режиме чтения записи.

```
# mkfs -t ext2 -c -c /dev/sdb5
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
125696 inodes, 251007 blocks
12550 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=260046848
8 block groups
32768 blocks per group, 32768 fragments per group
15712 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Testing with pattern 0xaa: done
Reading and comparing: done
Testing with pattern 0x55: done
Reading and comparing: done
Testing with pattern 0xff: done
Reading and comparing: done
Testing with pattern 0x00: done
Reading and comparing: done
```

```
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 21 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
#
```

Обратите внимание, на то, что в режиме проверки rw, происходит запись и считывание нескольких шаблонов. Это занимает довольно много времени.

Мы создали файловую систему *ext2*, но все же лучше использовать *ext3*. Это журналированная файловая система, более современная. Именно ее рекомендуют использовать в качестве основной файловой системы компания RedHat.

ext2 и *ext3* внутренне устроены абсолютно одинаково, единственное их различие — наличие поддержки журнала у *ext3*. Перевести *ext2* в *ext3* не составит никакого труда, достаточно создать файл журнала при помощи программы *tune2fs*.

```
# tune2fs -j /dev/sdb5
tune2fs 1.39 (29-May-2006)
Creating journal inode: done
This filesystem will be automatically checked every 21 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
#
```

Параметр *–j* заставляет программу создать файл журнала. В дальнейшем эту файловую систему можно использовать как *ext3*.

ПРОВЕРКА ФАЙЛОВОЙ СИСТЕМЫ.

Для проверки файловой системы используется программа *fsck*. По аналогии с программой *mkfs*, *fsck* — это программа оболочка, вызывающая для проверки специализированную для данной ФС программу.

В Linux рекомендуется проверять только не подключенные или подключенные в режиме только для чтения файловые системы. Проверка файловой системы, подключенной в режиме rw, может привести к порче этой ФС!

```
# fsck /dev/sdb5
fsck 1.39 (29-May-2006)
e2fsck 1.39 (29-May-2006)
/dev/sdb5: clean, 11/125696 files, 8367/251007 blocks
#
```

Программа выдала сообщение, что файловая система *clean*. Именно по этой причине не была произведена проверка ФС. В суперблоке есть специальное поле — состояние файловой системы, смотрите 8-ю строку вывода программы *dumpe2fs*:

```
# dumpe2fs /dev/sdb5 | head
dumpe2fs 1.39 (29-May-2006)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          b5013ca4-bffb-4188-8d7d-dc04dc0e02c0
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal resize_inode dir_index filetype sparse_super
large_file
```

```
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
#
```

Когда файловая система корректно отключается, в поле состояния ФС записывается clean. Если ФС выключить не корректно, то в этом поле будет другое состояние, и при старте системы будет запущена проверка.

Что бы принудительно проверить ФС, при вызове программы fsck необходимо использовать параметр *–f* (force).

```
# fsck -f /dev/sdb5
fsck 1.39 (29-May-2006)
e2fsck 1.39 (29-May-2006)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb5: 11/125696 files (9.1% non-contiguous), 8367/251007 blocks
#
```

ИСПОЛЬЗОВАНИЕ ФАЙЛОВЫХ СИСТЕМ.

Для того, что бы использовать файловую систему, ее необходимо подключить при помощи программы *mount*.

Файловая система подключается к любой существующей директории. Создадим в домашней директории точку монтирования.

```
# cd
# mkdir test
#
```

Теперь подключим файловую систему к созданной директории.

```
# mount /dev/sdb5 test
#
```

Список подключенных файловых систем можно посмотреть при помощи программы *mount*:

```
# mount
/dev/sda2 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda6 on /var type ext3 (rw)
/dev/sda5 on /tmp type ext3 (rw)
/dev/sda1 on /boot type ext3 (rw)
tmpfs on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/sdb5 on /root/test type ext3 (rw)
```

```
#
```

Или посмотрев содержимое файла */proc/mounts*.

```
# cat /proc/mounts
rootfs / rootfs rw 0 0
/dev/root / ext3 rw,data=ordered 0 0
/dev /dev tmpfs rw 0 0
/proc /proc proc rw 0 0
/sys /sys sysfs rw 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
devpts /dev/pts devpts rw 0 0
/dev/sda6 /var ext3 rw,data=ordered 0 0
/dev/sda5 /tmp ext3 rw,data=ordered 0 0
/dev/sda1 /boot ext3 rw,data=ordered 0 0
tmpfs /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
/etc/auto.misc /misc autofs
rw,fd=6,pgrp=2744,timeout=300,minproto=5,maxproto=5,indirect 0 0
-hosts /net autofs rw,fd=12,pgrp=2744,timeout=300,minproto=5,maxproto=5,indirect 0 0
/dev/sdb5 /root/test ext3 rw,data=ordered 0 0
#
```

Второй вариант более надежный, в этом случае мы смотрим содержимое оперативной памяти ядра. Программа *mount* смотрит содержимое файла */etc/mtab*. Это обыкновенный текстовый файл, который можно отредактировать и он может содержать не верную информацию.

Для отключения файловой системы используется программа *umount*.

```
# umount /root/test
#
```

В качестве параметра, программе можно указывать точку монтирования (директорию) или файл устройства.

ФАЙЛ /ETC/FSTAB.

Для того. чтобы после перезагрузки компьютера, файловые системы монтировались автоматически, необходимо добавить их описание в файл */etc/fstab*.

Формат файла очень простой. На одну файловую систему отводится одна строка. Поля в строке отделяются друг от друга любым количеством пробелов и/или табуляций.

- Первое поле — файл устройства или ключевое слово *LABEL*.
- Второе поле — точка монтирования.
- Третье поле — тип файловой системы. Тут можно написать несколько типов, через запятую или ключевое слово *auto* (имеет смысл для съемных накопителей).
- Четвертое поле — параметры монтирования.
- Пятое поле — флаг программы *dump*.
- Пятое поле — флаг программы *fsck*.

Рассмотрим поля подробнее. В первом поле нужно писать файл, где находится файловая система. В простейшем случае — это файл устройства. Например, */dev/sdb5*.

По умолчанию, программа установки дистрибутива каждому разделу присвоила *метку* (label). Если вы хотите указывать имя, тогда надо писать ключевое слово *LABEL=имя_метки*. Например, *LABEL=/boot*.

Во втором поле указывают директорию, к которой будет подключена файловая система. Если описывается swap пространство, в поле должно быть написано ключевое слово *swap*.

Тип файловой системы указывается в третьем поле. В простейшем случае тут будет записан тип. Но если вы описываете файловые системы съемных накопителей (DVD, flash и т.п.), где мы заранее не знаем, какой тип файловой системы будет на накопителе, вместо типа ФС можно писать ключевое слово *auto*, что бы программа попыталась сама определить тип.

К сожалению, такой вариант описание не всегда работает корректно. Поэтому вы можете перечислить несколько файловых систем, через запятую. При подключении устройства, программа сначала попробует использовать первую ФС. Если она не подойдет, тогда вторую и т.д.

Параметры монтирования, указываемые в четвертом поле, влияют на дальнейшую работу файловой системы. Какие параметры можно использовать зависит от типа подключаемой ФС. В документации к программе *mount* описаны почти все возможные параметры. Выделю только основные:

- ro — подключать файловую систему в режиме только для чтения.
- rw — подключать файловую систему в режиме полного доступа.
- noexec — не запускать исполняемые файлы, находящиеся в этой файловой системе.
- noatime — не изменять поле access time (время доступа) в файлах.
- nosuid — игнорировать специальные биты SUID, SGID и stiky в файловой системе.

Значение defaults в этом поле означает группу параметров:

- rw
- suid — включить действие специальных битов.
- dev — включить поддержку файлов устройств.
- exec — запускать на исполнение исполняемые файлы.
- auto — файловая система будет автоматически монтироваться, если при вызове программы mount указать параметр *–a*.
- nouser — простые пользователи не имеют права отключать и подключать эту файловую систему.

В других UNIX для бекапа файловых систем используют программу *dump*. К сожалению, в Linux из-за ряда причин, ее не рекомендуют использовать. Флаг оставлен в основном для совместимости. Если его значение равно нулю — файловая система не должна бекапиться автоматически. Если рано единице — ФС автоматически бекапится.

Флаг программы *fsck* используется аналогичным образом. Если значение поля равно нулю, файловая система не будет проверяться при старте компьютера. Если не равно нулю — будет проверяться.

Важно понимать, что файловые системы (кроме корневой) подключаются и проверяются в том порядке, в котором они описаны в этом файле.

Если мы хотим, чтобы файловая система */dev/sdb5* автоматически подключалась при старте компьютера, в файл */etc/fstab* необходимо добавить следующую строку:

```
/dev/sdb5               /root/test              ext3    defaults        0 1
```

После того как, вы описали файловую систему в */etc/fstab*, вы можете использовать сокращенный вариант вызова программы *mount*:

```
# mount /dev/sdb5
# mount
/dev/sda2 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda6 on /var type ext3 (rw)
/dev/sda5 on /tmp type ext3 (rw)
/dev/sda1 on /boot type ext3 (rw)
tmpfs on /dev/shm type tmpfs (rw)
```

```
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/sdb5 on /root/test type ext3 (rw)
#
```

Если при вызове программы *mount* указать не все параметры, она возьмет недостающие из конфигурационного файла */etc/fstab*.

Для того, что бы при старте компьютера автоматически подключалось дополнительное swap пространство, его тоже необходимо описать в */etc/fstab*.

Отключим файловую систему */dev/sdb5* и удалим соответствующую запись из файла */etc/fstab*.

```
# umount /dev/sdb5
#
```

СОЗДАНИЕ ПРОГРАММНОГО RAID.

При установке дистрибутива мы специально оставили нетронутым второй диск, для того, что бы в дальнейшем посмотреть как можно на лету организовать программный RAID1 в Linux.

Ниже приведен прядок действий, которые мы будем выполнять:

1. При помощи программы *sfdisk* создадим точную копию разделов на втором жестком диске.
2. Программой *fdisk* изменим тип разделов на втором диске на fd (Linux RAID).
3. Добавим разделы второго диска в RAID. Получим неполный (degraded) RAID.
4. Создадим файловые системы в разделе RAID.
5. Настроим систему для работы с RAID (файл */etc/fstab* и загрузчик *grub*).
6. Скопируем файлы из старой ФС в RAID.
7. Перезагрузим систему с использованием RAID.
8. Программой *fdisk* изменим типы разделов на первом диске.
9. Добавим разделы первого диска в RAID.

Перекрестимся, и потихоньку начнем работать.

Копирование разделов.

Скопируем разделы с диска */dev/sda* на */dev/sdb*. Для этого воспользуемся программой *sfdisk*.

```
# sfdisk -d /dev/sda | sfdisk /dev/sdb
Проверяется, чтобы сейчас никто не использовал этот диск...
OK

Диск /dev/sdb: 1827 цилиндров, 255 головок, 63 секторов/дорожку
Старая ситуация:
Единицы = цилиндры по 8225280 байт, блоки по 1024 байт, начиная с 0

  Устр-во Загр   Нач   Конец    #цил     #блоки   Id  Система
/dev/sdb1          0+     24      25-    200781   82  Linux своп / Solaris
/dev/sdb2         25    1826    1802   14474565    5  Расширенный
/dev/sdb3          0       -       0          0    0  Пустой
/dev/sdb4          0       -       0          0    0  Пустой
/dev/sdb5         25+    149     125-   1004031   83  Linux
Новая ситуация:
Единицы = секторы по 512 байт, начиная с 0
```

```
  Устр-во Загр    Начало      Конец    #секторы  Id  Система
/dev/sdb1   *         63     208844      208782  83  Linux
/dev/sdb2         208845   10442249    10233405  83  Linux
/dev/sdb3       10442250   12546764     2104515  82  Linux своп / Solaris
/dev/sdb4       12546765   29350754    16803990   5  Расширенный
/dev/sdb5       12546828   14651279     2104452  83  Linux
/dev/sdb6       14651343   29350754    14699412  83  Linux
Новая таблица разделов успешно записана

Перечитывается таблица разделов...

Если вы создали или изменили раздел DOS, скажем, /dev/foo7, используйте затем dd(1),
чтобы обнулить первые 512 байт:  dd if=/dev/zero of=/dev/foo7 bs=512 count=1
(См. fdisk(8).)
#
```

Учтите, что программа только изменила MBR на диске */dev/sdb*. Программа не переносит данные!

Проверим, какие разделы есть на наших дисках.

```
# fdisk -l

Диск /dev/sda: 15.0 ГБ, 15032385536 байт
255 heads, 63 sectors/track, 1827 cylinders
Единицы = цилиндры по 16065 * 512 = 8225280 байт

Устр-во Загр     Начало       Конец       Блоки   Id  Система
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14         650     5116702+  83  Linux
/dev/sda3             651         781     1052257+  82  Linux своп / Solaris
/dev/sda4             782        1827     8401995    5  Расширенный
/dev/sda5             782         912     1052226   83  Linux
/dev/sda6             913        1827     7349706   83  Linux

Диск /dev/sdb: 15.0 ГБ, 15032385536 байт
255 heads, 63 sectors/track, 1827 cylinders
Единицы = цилиндры по 16065 * 512 = 8225280 байт

Устр-во Загр     Начало       Конец       Блоки   Id  Система
/dev/sdb1   *           1          13      104391   83  Linux
/dev/sdb2              14         650     5116702+  83  Linux
/dev/sdb3             651         781     1052257+  82  Linux своп / Solaris
/dev/sdb4             782        1827     8401995    5  Расширенный
/dev/sdb5             782         912     1052226   83  Linux
/dev/sdb6             913        1827     7349706   83  Linux
#
```

Как видите, диски имеют одинаковое разбиение на разделы.

Изменение типа разделов.

Теперь на диске */dev/sdb* изменим типы разделов на *fd*. Это специальный тип, который будет использоваться в программном RAID. Для этого воспользуемся программой *fdisk*. Изменим все типы, кроме расширенного.

Обратите внимание на то, что и swap раздел необходимо оформить в виде RAID. Если этого не сделать, вы не сможете нормально загрузиться со второго диска, в случае порчи первого.

```
# fdisk /dev/sdb


Количество цилиндров для этого диска установлено в 1827.
С этим все в порядке, но значение больше, чем 1024,
и в отдельных установках могут возникнуть проблемы с:
1) программами, запускаемым при загрузке (напр., старые версии LILO)
2) загрузкой и программами разметки из других ОС
   (напр., DOS FDISK, OS/2 FDISK)


Команда (m для справки):
```

Для смены типа раздела используйте команду *t*.

```
Команда (m для справки): t
Номер раздела (1-6): 1
Шестнадцатеричный код (введите L для получения списка кодов): fd
Системный тип раздела 1 изменен на fd (Автоопределение Linux raid)


Команда (m для справки): p


Диск /dev/sdb: 15.0 ГБ, 15032385536 байт
255 heads, 63 sectors/track, 1827 cylinders
Единицы = цилиндры по 16065 * 512 = 8225280 байт


Устр-во Загр     Начало       Конец       Блоки   Id  Система
/dev/sdb1   *           1          13      104391   fd  Автоопределение Linux raid
/dev/sdb2              14         650     5116702+  83  Linux
/dev/sdb3             651         781     1052257+  82  Linux своп / Solaris
/dev/sdb4             782        1827     8401995    5  Расширенный
/dev/sdb5             782         912     1052226   83  Linux
/dev/sdb6             913        1827     7349706   83  Linux


Команда (m для справки):
```

После ввода команды *t* нас попросили ввести номер изменяемого раздела. Вводим *1* и тип раздела *fd*. При помощи команды *p* просматриваем список разделов и убеждаемся, что раздел номер 1, поменял свой тип.

Аналогичным образом изменим типы у всех разделов, кроме расширенного.

```
Команда (m для справки): t
Номер раздела (1-6): 2
Шестнадцатеричный код (введите L для получения списка кодов): fd
Системный тип раздела 2 изменен на fd (Автоопределение Linux raid)
```

```
Команда (m для справки): t
Номер раздела (1-6): 3
Шестнадцатеричный код (введите L для получения списка кодов): fd
Системный тип раздела 3 изменен на fd (Автоопределение Linux raid)

Команда (m для справки): t
Номер раздела (1-6): 5
Шестнадцатеричный код (введите L для получения списка кодов): fd
Системный тип раздела 5 изменен на fd (Автоопределение Linux raid)

Команда (m для справки): t
Номер раздела (1-6): 6
Шестнадцатеричный код (введите L для получения списка кодов): fd
Системный тип раздела 6 изменен на fd (Автоопределение Linux raid)

Команда (m для справки): p

Диск /dev/sdb: 15.0 ГБ, 15032385536 байт
255 heads, 63 sectors/track, 1827 cylinders
Единицы = цилиндры по 16065 * 512 = 8225280 байт

Устр-во Загр     Начало       Конец       Блоки   Id  Система
/dev/sdb1   *           1          13      104391   fd  Автоопределение Linux raid
/dev/sdb2              14         650     5116702+  fd  Автоопределение Linux raid
/dev/sdb3             651         781     1052257+  fd  Автоопределение Linux raid
/dev/sdb4             782        1827     8401995    5  Расширенный
/dev/sdb5             782         912     1052226   fd  Автоопределение Linux raid
/dev/sdb6             913        1827     7349706   fd  Автоопределение Linux raid

Команда (m для справки):
```

При помощи команды *w* сохраним изменения и выйдем из программы.

```
Команда (m для справки): w
Таблица разделов была изменена!

Вызывается ioctl() для перечитывания таблицы разделов.
Синхронизируются диски.
#
```

Создание RAID массивов.

Теперь создадим несколько RAID массивов. В каждый массив поместим один раздел диска.

```
# mdadm --create /dev/md0 --level=1 --raid-disks=2 missing /dev/sdb1
mdadm: array /dev/md0 started.
# mdadm --create /dev/md1 --level=1 --raid-disks=2 missing /dev/sdb2
mdadm: array /dev/md1 started.
# mdadm --create /dev/md2 --level=1 --raid-disks=2 missing /dev/sdb3
```

```
mdadm: array /dev/md2 started.
# mdadm --create /dev/md3 --level=1 --raid-disks=2 missing /dev/sdb5
mdadm: array /dev/md3 started.
# mdadm --create /dev/md4 --level=1 --raid-disks=2 missing /dev/sdb6
mdadm: array /dev/md4 started.
#
```

Утилита *mdadm* применяется для управления программным RAID. В нашем случае мы указали команду *--create*, для создания раздела.

Затем мы определили файл устройства */dev/md*.

При помощи параметра *--level=1* мы сказали, что будем делать RAID 1.

Параметр *--raid-disks=2* говорит, что в RAID будут использоваться два диска.

В самом конце мы определяем файлы устройств, которые будут использоваться в RAID. Вместо первого диска мы написали ключевое слово *missing*. Это означает, что первого диска в RAID пока нет.

В результате мы получаем неполные RAID1.

Убедимся, что RAID работает:

```
# cat /proc/mdstat
Personalities : [raid1]
md4 : active raid1 sdb6[1]
      7349632 blocks [2/1] [_U]

md3 : active raid1 sdb5[1]
      1052160 blocks [2/1] [_U]

md2 : active raid1 sdb3[1]
      1052160 blocks [2/1] [_U]

md1 : active raid1 sdb2[1]
      5116608 blocks [2/1] [_U]

md0 : active raid1 sdb1[1]
      104320 blocks [2/1] [_U]

unused devices: <none>
#
```

Статус *_U* говорит, что первого диска в RAID нет, а второй диск работает (up).

Теперь в каждом разделе RAID создадим файловые системы.

```
# mkfs -t ext3 /dev/md0
# mkfs -t ext3 /dev/md1
# mkfs -t ext3 /dev/md3
# mkfs -t ext3 /dev/md4
```

И swap пространство.

```
# mkswap /dev/md2
```

Создадим временные директории, к которым мы подключим файловый системы RAID.

```
# mkdir /mnt/{md0,md1,md3,md4}
# mount /dev/md0 /mnt/md0
# mount /dev/md1 /mnt/md1
# mount /dev/md3 /mnt/md3
# mount /dev/md4 /mnt/md4
#
```

Создадим конфигурационный файл программы *mdadm* — */etc/mdadm.conf*. И добавим в него всего одну строку: *DEVICE partitions*.

```
# echo "DEVICE partitions" > /etc/mdadm.conf
#
```

Занесем информацию о RAID массивах в этот файл.

```
# mdadm --detail --scan >> /etc/mdadm.conf
```

Посмотрим содержимое файла.

```
# cat /etc/mdadm.conf
DEVICE partitions
ARRAY /dev/md0 level=raid1 num-devices=2 UUID=cd440d9f:a0eef4be:eea538a2:cc58071f
ARRAY /dev/md1 level=raid1 num-devices=2 UUID=fa5b24e7:a4bc0d07:4705f4a2:61d47277
ARRAY /dev/md2 level=raid1 num-devices=2 UUID=7f6c4a1d:ecaa66fb:ffe104b5:0b76d7db
ARRAY /dev/md3 level=raid1 num-devices=2 UUID=7e3e4f75:e28e3ea7:41ad7318:8dc28b77
ARRAY /dev/md4 level=raid1 num-devices=2 UUID=b7613db8:c568ade5:9ac49918:c562f6d7
#
```

Настройка загрузчика.

Изменим содержимое файла */etc/fstab*, заменив текущие устройства на устройства RAID. Это необходимо сделать по двум причинам:

1. Что бы в дальнейшем при загрузке использовались RAID массивы.
2. Что бы при создании образа загрузочного RAM диска, программа mkinitrd поместила в образ необходимые драйвера.

После внесения изменений должен получиться файл следующего содержания:

```
/dev/md1                /                       ext3    defaults        1 1
/dev/md4                /var                    ext3    defaults        1 2
/dev/md3                /tmp                    ext3    defaults        1 2
/dev/md0                /boot                   ext3    defaults        1 2
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
/dev/md2                swap                    swap    defaults        0 0
```

Редактируем файл */boot/grub/device.map*. Добавляем устройство *hd1*.

```
(hd0)   /dev/sda
(hd1)   /dev/sdb
```

Теперь отредактируем конфигурационный файл загрузчика */boot/grub/grub.conf*.

Сразу после строки:

```
hiddenmenu
```

Добавьте строки, описывающие загрузку с разных жестких дисков.

```
title CentOS (2.6.18-92.el5) RAID1 sda
        root (hd0,0)
        kernel /vmlinuz-2.6.18-92.el5 ro root=/dev/md1
        initrd /initrd-2.6.18-92.el5.img
title CentOS (2.6.18-92.el5) RAID1 sdb
        root (hd1,0)
        kernel /vmlinuz-2.6.18-92.el5 ro root=/dev/md1
        initrd /initrd-2.6.18-92.el5.img
```

Обратите внимание на команду *root (hd0,0)* и *root (hd1,1)*. А так же на параметр ядра *root=/dev/md0*.

Создадим образ базового RAM диска. Его необходимо сделать до того, как мы начнем копировать данные!

```
# mkinitrd /boot/initrd-`uname -r`-raid1.img `uname -r`
#
```

Обратите внимание на то, что программа *uname* помещена в обратные одинарные кавычки.

Скопируем содержимое файловых систем на диске *sda* соответствующие разделы RAID. Копировать будем файлы, при помощи стандартной программы *cp*.

Процедура копирования файлов займет много времени из-за того, что наш RAID массив организован на базе виртуальных винтов виртуальной машины. В реальных системах, с разнесенными по разным устройствам RAID будет работать гораздо быстрее.

```
# cp -dpRx / /mnt/md1
# cp -dpRx /var/* /mnt/md4
# cp -dpRx /tmp/* /mnt/md3
# cp -dpRx /boot/* /mnt/md0
```

Программа cp скопирует содержимое директорий, сохраняя все права доступа.

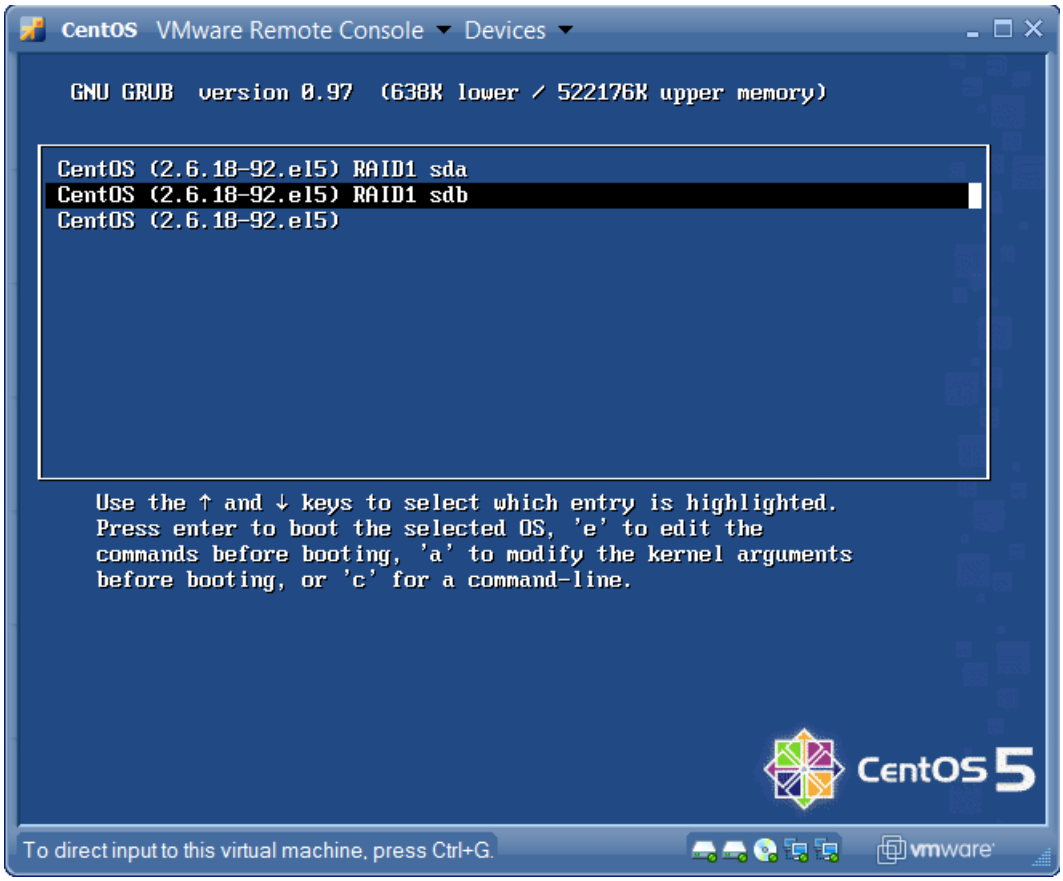
Теперь настроим загрузчик grub.

Запускаем программу *grub* и в ее командной строке вводим команды в следующем листинге выделенные подчеркиванием:

```
grub> root (hd0,0)
 Filesystem type is ext2fs, partition type 0x83


grub> setup (hd0)
 Checking if "/boot/grub/stage1" exists... no
 Checking if "/grub/stage1" exists... yes
 Checking if "/grub/stage2" exists... yes
 Checking if "/grub/e2fs_stage1_5" exists... yes
 Running "embed /grub/e2fs_stage1_5 (hd0)"...  15 sectors are embedded.
succeeded
 Running "install /grub/stage1 (hd0) (hd0)1+15 p (hd0,0)/grub/stage2 /grub/grub
.conf"... succeeded
Done.


grub> root (hd1,0)
```

```
 Filesystem type is ext2fs, partition type 0xfd

grub> setup (hd1)
 Checking if "/boot/grub/stage1" exists... no
 Checking if "/grub/stage1" exists... yes
 Checking if "/grub/stage2" exists... yes
 Checking if "/grub/e2fs_stage1_5" exists... yes
 Running "embed /grub/e2fs_stage1_5 (hd1)"...  15 sectors are embedded.
succeeded
 Running "install /grub/stage1 (hd1) (hd1)1+15 p (hd1,0)/grub/stage2 /grub/grub
.conf"... succeeded
Done.

grub> quit
```

Перезагружаемся, при перезагрузке выбираем второе ядро.



Если по каким то причинам, загрузиться не удается, перегрузите виртуальную машину, и при загрузке выберите старый вариант - последний пункт. Обратитесь за помощью к преподавателю.

Добавление первого диска в RAID массив.

Меняем тип разделов на диске *sda*, при помощи программы *fdisk*, точно так же как мы это делали на устройстве *sdb*.

Добавляем в RAID устройства.

```
# mdadm --add /dev/md0 /dev/sda1
```

```
mdadm: added /dev/sda1
# mdadm --add /dev/md1 /dev/sda2
mdadm: added /dev/sda2
# mdadm --add /dev/md2 /dev/sda3
mdadm: added /dev/sda3
# mdadm --add /dev/md3 /dev/sda5
mdadm: added /dev/sda5
# mdadm --add /dev/md4 /dev/sda6
mdadm: added /dev/sda6
```

После этого вы можете понаблюдать, как происходит синхронизация устройств.

```
# watch cat /proc/mdstat
```

Программа *watch* будет раз в две секунды запускать программу *cat* и вы почти в реальном времени сможете наблюдать за процессом синхронизации.

```
Every 2,0s: cat /proc/mdstat


Personalities : [raid1]
md1 : active raid1 sdb2[1]
      5116608 blocks [2/1] [_U]

md2 : active raid1 sda3[0] sdb3[1]
      1052160 blocks [2/2] [UU]

md3 : active raid1 sda5[2] sdb5[1]
      1052160 blocks [2/1] [_U]
        resync=DELAYED

md4 : active raid1 sda6[2] sdb6[1]
      7349632 blocks [2/1] [_U]
      [>....................]  recovery =  0.1% (10688/7349632) finish=34.2min s
peed=3562K/sec

md0 : active raid1 sda1[0] sdb1[1]
      104320 blocks [2/2] [UU]

unused devices: <none>
```

Процедура синхронизации может занять много времени.

В результате мы получаем следующую картину.

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdb1[1] sda1[0]
      104320 blocks [2/2] [UU]

md2 : active raid1 sdb3[1] sda3[0]
      1052160 blocks [2/2] [UU]
```

```
md3 : active raid1 sdb5[1] sda5[0]
      1052160 blocks [2/2] [UU]

md4 : active raid1 sdb6[1] sda6[0]
      7349632 blocks [2/2] [UU]

md1 : active raid1 sda2[0] sdb2[1]
      5116608 blocks [2/2] [UU]

unused devices: <none>
#
```

Откройте в редакторе конфигурационный файл */etc/mdadm.conf* и удалите строки, описывающие устройства входящие в RAID. Сохраните файл. Заполните его новой информацией.

```
# mdadm --detail --scan >> /etc/mdadm.conf
#
```

Обязательно пересоберите образ первоначального RAM диска!

```
# mkinitrd -f /boot/initrd-`uname -r`-raid1.img `uname -r`
#
```

Перезагружаемся и выбираем первое ядро для загрузки.