

```
user@centos8:~$ ls /dev/
autofs          lp0             shm             tty26           tty50           usbmon2
block           lp1             snapshot       tty27           tty51           vboxguest
bsg             lp2             snd            tty28           tty52           vboxuser
bus            lp3             sr0            tty29           tty53           vcs
cdrom          mapper          stderr         tty3            tty54           vcs1
char           mcelog          stdin          tty30           tty55           vcs2
cl_centos8      mem            stdout         tty31           tty56           vcs3
console        memory_bandwidth tty             tty32           tty57           vcs4
core           mqueue         tty0           tty33           tty58           vcs5
cpu            net            tty1           tty34           tty59           vcs6
cpu_dma_latency network_latency tty10          tty35           tty6            vcsa
disk           network_throughput tty11          tty36           tty60           vcsa1
dm-0           null           tty12          tty37           tty61           vcsa2
dm-1           nvram          tty13          tty38           tty62           vcsa3
```

В прошлый раз мы разобрались, что udev при виде определённых устройств создаёт для них специальные файлы в директории /dev - ls /dev, через которые можно взаимодействовать с устройствами. И если с большинством устройств администратору не требуется ничего делать, то вот с устройствами хранения информации практически постоянно нужно работать. При работе с дисками нужно быть крайне осторожными, потому что на дисках данные, а какие-то ошибки могут привести к потере этих данных. Поэтому всегда делайте бэкапы и убеждайтесь, что они в рабочем состоянии.



Есть разные типы накопителей и они могут по разному подключаться к компьютеру – старые диски подключались по IDE, сейчас преимущество у SATA, набирают популярность nvme SSD, есть ещё всякие флешки, подключаемые по usb, во многих компаниях диски выдаются по сети хранения данных, называемой SAN, а в облачных средах вам выдаются виртуальные диски, детали подключения которых вас могут и не интересовать. В зависимости от некоторых факторов – типа подключения диска, правил udev, которые могут отличаться в зависимости от дистрибутива – дискам могут выдаваться различные названия. При этом, несмотря на различия, для работы с устройствами хранения используется протокол SCSI – это касается и USB флешек, и SATA дисков, и сетей хранения данных и многого другого.

```
[user@centos8 ~]$ ls /dev/sd*
/dev/sda /dev/sda1 /dev/sda2
[user@centos8 ~]$ ls /dev/sr*
/dev/sr0
[user@centos8 ~]$ ls -l /dev/sda
brw-rw----. 1 root disk 8, 0 Feb 14 15:16 /dev/sda
[user@centos8 ~]$ stat /dev/sda
  File: /dev/sda
  Size: 0          Blocks: 0          IO Block: 4096   block special file
Device: 6h/6d  Inode: 12991      Links: 1        Device type: 8,0
Access: (0660/brw-rw----)  Uid: (  0/   root)   Gid: (  6/   disk)
Context: system_u:object_r:fixed_disk_device_t:s0
Access: 2021-02-14 15:16:13.735000259 +0400
Modify: 2021-02-14 15:16:12.652000225 +0400
Change: 2021-02-14 15:16:12.652000225 +0400
 Birth: -
[user@centos8 ~]$
```

Поэтому большая часть накопителей будет именоваться sd – т.е. scsi диск - ls /dev/sd* - а дальше каждому устройству будет даваться буква по алфавиту – sda, sdb, sdc и т.п. Для cd приводов будет даваться название sr - ls /dev/sr* - sr0, sr1 и т.п. А в облаках вы будете наткаться на названия vda, vdb и т.д. Что объединяет все накопители? Операционная система обменивается с этими устройствами данными в виде блоков данных фиксированной длины. Команда ls -l - ls -l /dev/sda, stat /dev/sda - покажет перед правами символ b – указывающий, что это блочное устройство.

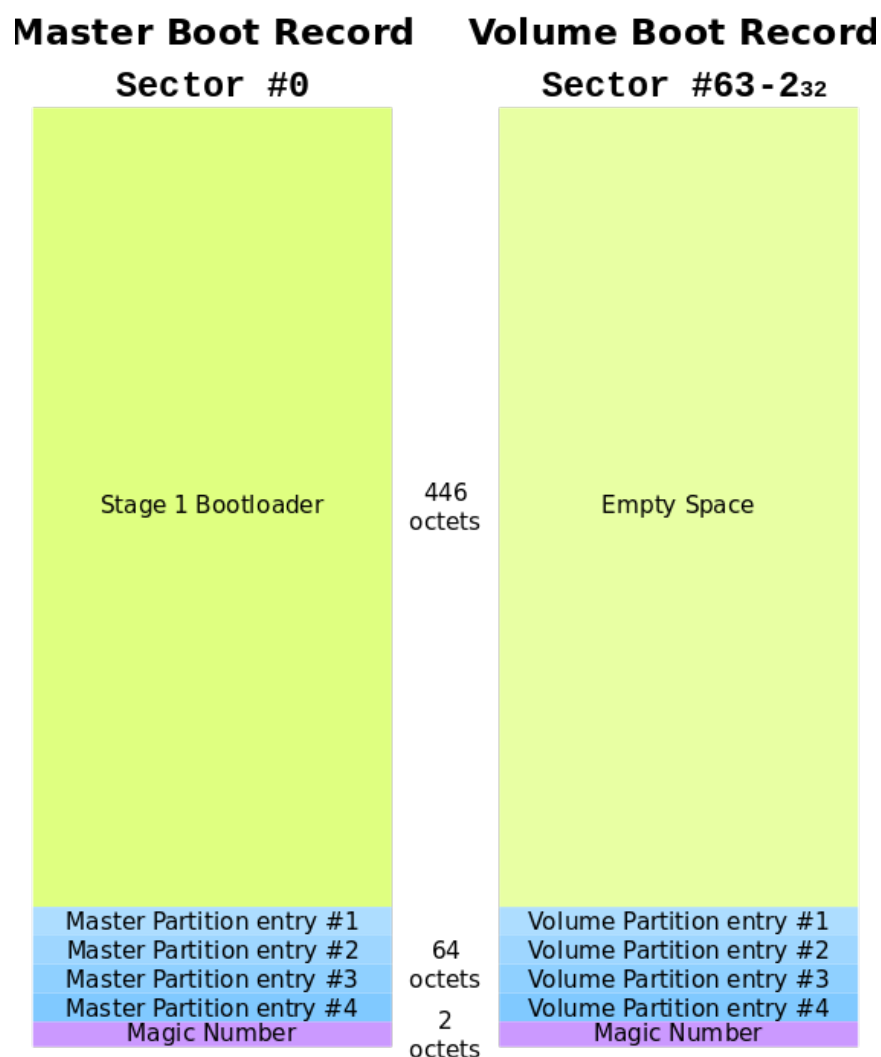
```
[user@centos8 ~]$ ls -l /dev/input/mouse0
crw-rw----. 1 root input 13, 32 Feb 14 15:16 /dev/input/mouse0
[user@centos8 ~]$ stat /dev/input/mouse0
  File: /dev/input/mouse0
  Size: 0          Blocks: 0          IO Block: 4096   character special file
Device: 6h/6d  Inode: 11121      Links: 1        Device type: d,20
Access: (0660/crw-rw----)  Uid: (  0/   root)   Gid: (999/   input)
Context: system_u:object_r:mouse_device_t:s0
Access: 2021-02-14 15:16:12.468000219 +0400
Modify: 2021-02-14 15:16:12.468000219 +0400
Change: 2021-02-14 15:16:12.468000219 +0400
 Birth: -
[user@centos8 ~]$
```

Кроме блочных устройств существуют символьные – такие устройства работают с потоком данных, а не с блоками. Допустим, та же мышка - ls -l /dev/input/mouse0, stat /dev/input/mouse0. Перед такими файлами стоит символ c – character device – символьное устройство.

```
[user@centos8 ~]$ lsscsi -s
[1:0:0:0]    cd/dvd  VBOX    CD-ROM    1.0    /dev/sr0    -
[2:0:0:0]    disk    ATA     VBOX HARDDISK  1.0    /dev/sda    21.4GB
[user@centos8 ~]$
```

Но, как я уже сказал, чаще всего накопители работают через scsi протоколы – поэтому, как бы там не было написано в правилах udev, нам необязательно гадать – мы можем с помощью утилиты lsscsi увидеть наши диски - lsscsi -s. Как видите, тут у меня подключены дисковод и диск на 20 гигабайт, который получил название sda. Но названия, которые даёт udev – sda, sdb и т.п. - не закрепляются за дисками навсегда. Каждый раз, когда вы запускаете систему или подключаете устройства, udev даёт название по порядку. Да, есть порядок обнаружения устройств и зачастую одни и те же диски будут называться одинаково, но ни в коем случае нельзя ориентироваться на эти названия. Допустим, если у вас 3 диска – sda, sdb и sdc и вы переподключите их, либо один перестанет работать – то sdc начнёт называться sdb. В теории это может привести к потере данных. Как? Мы разберём чуть позже. Просто запомните, что ориентироваться на эти буквы не стоит.

Из темы “О файловых системах” мы выяснили - чтобы мы могли создавать, хранить, изменять и в целом работать с файлами на диске - нам нужна файловая система. Для этого мы можем записать её на диск. В принципе, такая схема будет работать, но в целом это неудобно и может создать нам проблемы в будущем. Например, в будущем нам может понадобится переустановить систему. Как правило, это предполагает удаление старой файловой системы и создание новой – это называется форматированием файловой системы. Но, при этом, все файлы, которые мы хотели бы перенести на новую систему, также затрут. Конечно, можно заранее закинуть всё на флешку и потом вернуть обратно – но это лишняя работа и потеря времени. Возможно, вы знаете как избежать этой проблемы – на том же Windows у вас может быть том D, на который вы кидаете файлы, и при форматировании эти данные не стираются – потому что стирается файловая система в том C. На GNU/Linux файлы пользователей хранятся в директории /home, поэтому вам будет достаточно отделить /home от корня. Т.е. предполагается, что у вас две файловые системы на одном диске. Для этого нужно разделить диск на так называемые разделы, и на каждый раздел записать свою файловую систему. Но чтобы компьютер знал – где начинается один раздел, где он заканчивается и начинается другой – нужно специальное место в начале диска, где указывается эта информация – таблица разделов.



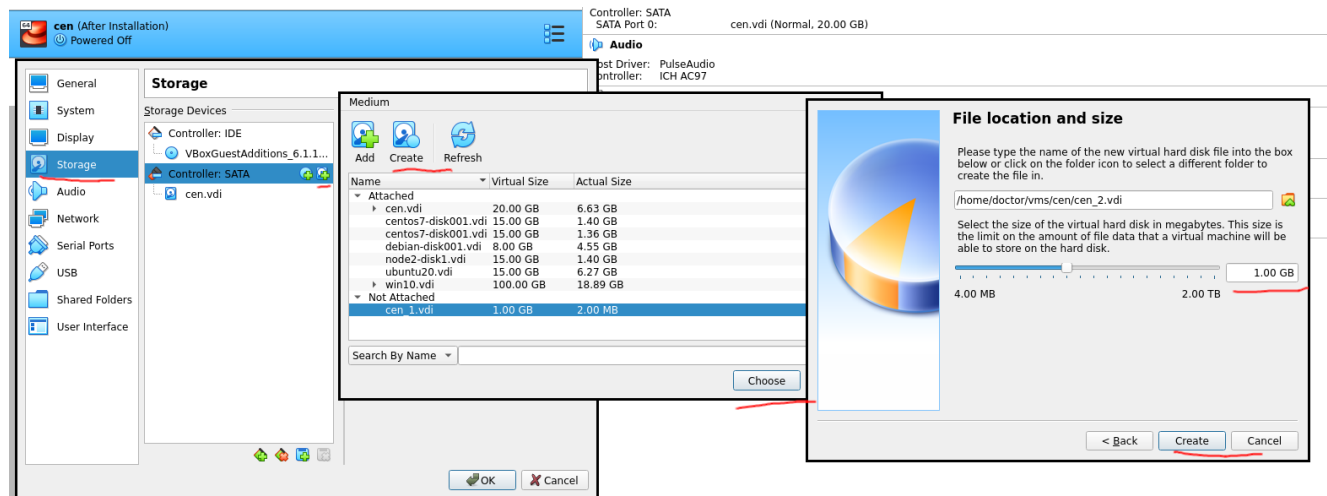
Each partition table entry comprises of 16 octets:

Flag	Start CHS	Type	End CHS	Start LBA	Size
1	3	1	3	4	4 octets

Есть разные типы таблиц разделов: MBR – которую также называют dos или ms-dos; GPT; у Apple и BSD свои таблицы разделов; есть ещё какие-то – но это нас не интересует, в основном вы будете иметь дело с MBR и GPT. У аббревиатуры MBR есть и другое значение – главная загрузочная запись – и сама таблица разделов хранится внутри этой записи. А загрузочная запись MBR была нужна для компьютеров раньше. Дело в том, что раньше на компьютерах был чип BIOS, в котором был ряд микропрограмм, и, кроме всего прочего, BIOS отвечал за включение компьютера. Но BIOS был сильно ограничен – он должен был быть не больше десятка килобайт, оперативки ему было доступно было максимум мегабайт – ну и в таких условиях сильно не разгуляешься. И BIOS должен был в итоге загрузить операционную систему – но ведь с такими ограничениями не добавить поддержку какой-то файловой системы и программы, загружающей операционную систему. Причём операционные системы то разные, у каждого своя файловая система, каждую по своему грузить. Поэтому BIOS просто обращался к нулевому сектору жёсткого диска, где и находилась главная загрузочная запись - MBR. А там у нас и загрузчик операционной системы и таблица разделов. При этом сам MBR тоже был ограничен – всего 512 байт, из которых 446 байт на загрузчик и 64 на таблицу разделов. Забегая вперёд, скажу, что там ещё первые 63 сектора оставались свободными, в которые и помещается

основная часть загрузчика, а не только эти 446 байт, но это тема загрузки операционной системы, мы это рассмотрим в другой раз.

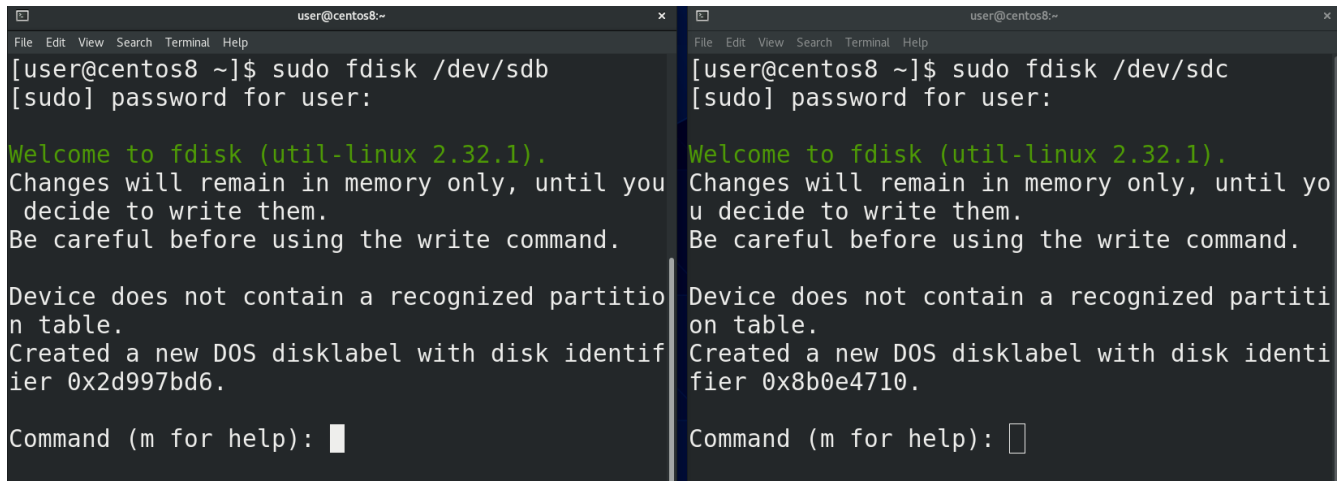
Так вот, MBR нам даёт 64 байта на то, чтобы поместить всю информацию о разделах. А на информацию о каждом разделе требуется 16 байт. В итоге – 4 раздела. Потом появилась расширенная загрузочная запись (VBR), благодаря чему можно было создавать расширенные разделы, которые позволяли обойти ограничение в 4 раздела. Грубо говоря, там вместо самой записи о разделе указывалась ссылка на другую таблицу, в которой указывались дополнительные разделы. Также в таблице разделов MBR не получилось бы указать разделы больше 2Тб, так как адрес не помещается в таблице. По итогу, BIOS своими ограничениями создавал кучу проблем с вынужденными обходными решениями. В нулевых же решили избавиться от этих ограничений и создали новый стандарт – UEFI. Тут уже разгулялись – UEFI может весить десяток мегабайт, понимает файловые системы, может работать с сетью, имеет графический интерфейс. Точнее, это всё можно реализовать в UEFI, но не каждый производитель это делает, разве что какой-то стандартный функционал. И, конечно же, отпала необходимость в загрузчике в MBR – зачем ограничиваться 446 байтами, когда можно в UEFI добавить поддержку файловой системы, где и будет лежать полноценный загрузчик? Плюс растут объёмы дисков – поэтому MBR заменили на GPT. Необходимость держать загрузчик в начальном секторе отпала – UEFI для этого использует специальный раздел EFI с файловой системой FAT32. Ограничение в 2 Тб тоже пропало, у GPT теоретическое ограничение почти в 10 Зеттабайт. Что касается количества разделов – то они, в принципе, не ограничены, разве что только со стороны операционной системы, но там речь про 128 разделов, чего с лихвой хватает. Но, при этом, UEFI в целях совместимости позволяет установить загрузчик в нулевой сектор, как это было в MBR.



```
[user@centos8 ~]$ lsblk -s
[1:0:0:0]    cd/dvd  VBOX    CD-ROM    1.0    /dev/sr0    -
[2:0:0:0]    disk    ATA     VBOX HARDDISK  1.0    /dev/sda    21.4GB
[3:0:0:0]    disk    ATA     VBOX HARDDISK  1.0    /dev/sdb    1.07GB
[4:0:0:0]    disk    ATA     VBOX HARDDISK  1.0    /dev/sdc    1.07GB
```

Прежде чем пойдём дальше, давайте немного попрактикуемся с таблицами разделов. Но для начала нам понадобится добавить диски для нашей виртуальной машины. Поэтому выключаем виртуалку, открываем её настройки, переходим во вкладку Storage – Controller SATA

и добавляем два жёстких диска, создаём их – по гигабайту будет достаточно, можно даже обойтись меньшим объёмом. Нажимаем ОК и запускаем виртуалку. Теперь ls SCSI -s покажет нам ещё два диска по гигабайту, с названиями sdb и sdc.



The image shows two side-by-side terminal windows. The left window is titled 'user@centos8:~' and shows the command 'sudo fdisk /dev/sdb' being executed. The output indicates that the device does not contain a recognized partition table and that a new DOS disklabel with disk identifier 0x2d997bd6 has been created. The right window is also titled 'user@centos8:~' and shows the command 'sudo fdisk /dev/sdc' being executed. The output is similar, indicating that a new DOS disklabel with disk identifier 0x8b0e4710 has been created. Both windows show the 'Command (m for help):' prompt.

```
[user@centos8 ~]$ sudo fdisk /dev/sdb
[sudo] password for user:

Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you
decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition
table.
Created a new DOS disklabel with disk identifi
er 0x2d997bd6.

Command (m for help):
```

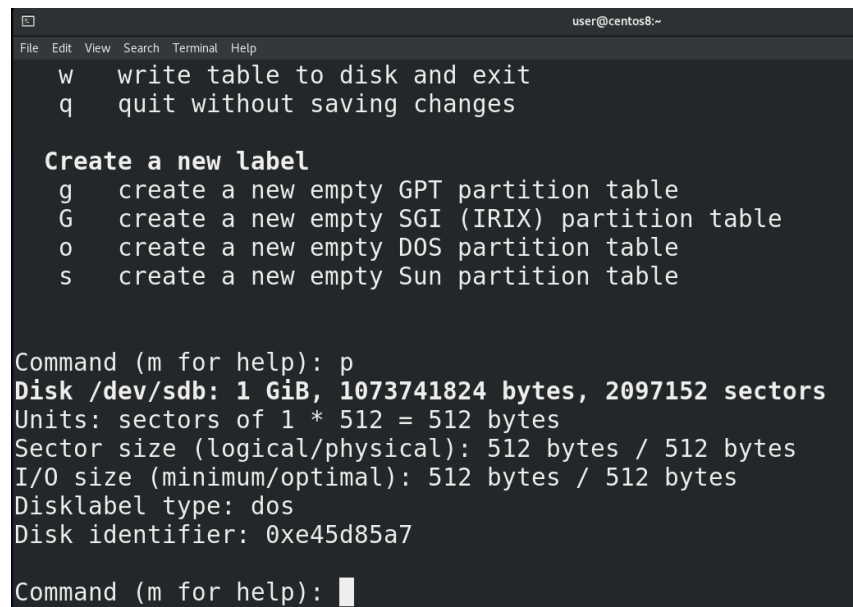
```
[user@centos8 ~]$ sudo fdisk /dev/sdc
[sudo] password for user:

Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until yo
u decide to write them.
Be careful before using the write command.

Device does not contain a recognized partiti
on table.
Created a new DOS disklabel with disk identi
fier 0x8b0e4710.

Command (m for help):
```

И так, диски у нас есть – sdb и sdc. Давайте создадим таблицу разделов и пару разделов. Для этого используем утилиту fdisk - sudo fdisk /dev/sdb; sudo fdisk /dev/sdc. И так, для начала, нас предупредили, что всё что мы делаем – не сразу происходит, а сохраняется в памяти, и если мы захотим – то пишем w и изменения вступают в силу. Далее нас предупредили, что на диске не нашли никакой таблицы разделов, поэтому программа создала таблицу DOS – но опять же, без write-а ничего на деле не изменилось.



The image shows a terminal window titled 'user@centos8:~'. It displays the fdisk utility for disk /dev/sdb. The output shows the disk size (1 GiB, 1073741824 bytes, 2097152 sectors) and the disklabel type (dos). The disk identifier is 0xe45d85a7. The prompt 'Command (m for help):' is shown at the bottom.

```
user@centos8:~
File Edit View Search Terminal Help

w  write table to disk and exit
q  quit without saving changes

Create a new label
g  create a new empty GPT partition table
G  create a new empty SGI (IRIX) partition table
o  create a new empty DOS partition table
s  create a new empty Sun partition table

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe45d85a7

Command (m for help):
```

Ну и небольшая подсказка по командам с помощью m. Для начала выведем информацию – буква p. Наш диск – /dev/sdb, его размер – 1 Гиббайт, количество байт и секторов. Каждый сектор по 512 байт – умножаем на количество секторов – получаем количество байт. Буква o – создаёт таблицу разделов DOS, буква g – GPT.

```

Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151): +200M

Created a new partition 1 of type 'Linux' and of size 200 MiB.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot Start    End Sectors  Size Id Type
/dev/sdb1                2048 411647  409600   200M 83 Linux

```

И так, с помощью `n` создали таблицу разделов, теперь можем создавать разделы. Для этого буква `n` – new partition. В случае с DOS у нас спрашивает – будет ли это основной раздел или расширенный. Основных может быть 4, а для создания расширенного используется один из основных. Выберем основной – по умолчанию он и создаётся, поэтому просто `enter`. Далее нужно выбрать его номер – допустим 1. Потом выбираем начальный сектор – пусть будет 2048 – оставляем по умолчанию. Далее пишем последний сектор, либо необходимый размер для раздела через `+`, допустим `+200M`. Создалось. Теперь пишем `p` и видим новый раздел. Разделы называются как соответствующий диск – то есть `sdb`, а в конце указывается номер раздела - `sdb1`.


```

Command (m for help): n
Partition type
  p   primary (1 primary, 0 extended, 3 free)
  e   extended (container for logical partitions)
Select (default p): e
Partition number (2-4, default 2): 4
First sector (411648-2097151, default 411648):
Last sector, +sectors or +size{K,M,G,T,P} (411648-2097151, default 2097151):

Created a new partition 4 of type 'Extended' and of size 823 MiB.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot  Start      End  Sectors  Size Id Type
/dev/sdb1                2048  411647   409600  200M 83 Linux
/dev/sdb4                411648 2097151 1685504   823M  5 Extended

```

Давайте, для примера, создадим ещё один раздел. n - выберем extended – в качестве основного используем 4; первый сектор оставим как есть – это первый незанятый сектор; дальше просто нажмём enter – тогда используется всё доступное пространство. Опять p – видим наши разделы. Но сам по себе расширенный раздел мы использовать не будем – он просто позволит создавать внутри него другие разделы, называемые логическими разделами.

```

Command (m for help): n
All space for primary partitions is in use.
Adding logical partition 5
First sector (413696-2097151, default 413696):
Last sector, +sectors or +size{K,M,G,T,P} (413696-2097151, default 2097151): +100M

Created a new partition 5 of type 'Linux' and of size 100 MiB.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot  Start      End  Sectors  Size Id Type
/dev/sdb1                2048  411647   409600  200M 83 Linux
/dev/sdb4                411648 2097151 1685504   823M  5 Extended
/dev/sdb5                413696  618495   204800   100M 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.

```

Поэтому опять n – оставляем первый сектор как есть, потом указываем размер раздела - +100M, и опять p. Теперь видим, что у нас есть 3 раздела: первый - sdb1 – это основной раздел;

дальше у нас расширенный раздел – sdb4 – он у нас вроде платформы, внутри которой мы создаём логические разделы, к примеру - sdb5. Когда нас всё устраивает, мы пишем w и изменения сохраняются.

```
Command (m for help): g
Created a new GPT disklabel (GUID: 02541524-4E41-2846-AD26-7B9A332308F9).

Command (m for help): n
Partition number (1-128, default 1):
First sector (2048-2097118, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097118, default 2097118): +200M

Created a new partition 1 of type 'Linux filesystem' and of size 200 MiB.

Command (m for help): p
Disk /dev/sdc: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 02541524-4E41-2846-AD26-7B9A332308F9

Device      Start      End Sectors  Size Type
/dev/sdc1   2048 411647 409600   200M Linux filesystem

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
```

Теперь создадим разделы на диске sdc с таблицей разделов GPT. Для её создания - g. Для нового раздела – n. У нас спрашивают номер раздела – оставим 1. Первый сектор – оставляем как есть. Последний сектор - +200M. И p, чтобы вывести информацию о разделах. Как видите, никакой возни с расширенными и логическими разделами. Опять же - w – сохраняем изменения.

```
Command (m for help): d
Partition number (1,4,5, default 5): 4

Partition 4 has been deleted.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot Start      End Sectors  Size Id Type
/dev/sdb1                   2048 411647 409600   200M 83 Linux

Command (m for help):
```

Для удаления какого-то раздела - sudo fdisk /dev/sdb – буква d – выбираем раздел – 4, enter, а потом p. Как видите, sdb5, который был внутри sdb4, удалился в том числе. Поэтому просто нажимаем q и ничего не сохраняем.

```
[user@centos8 ~]$ sudo fdisk -l /dev/sda
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xae357d6b
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sda1	*	2048	2099199	2097152	1G	83	Linux
/dev/sda2		2099200	41943039	39843840	19G	8e	Linux LVM

Чтобы увидеть, что у нас с разделами в системе – `sudo fdisk -l`. Тут у нас и то, что мы создали только что, а также `sda1` и `sda2` - там где у нас хранится система. Если посмотреть внимательнее на `sda2` - `sudo fdisk -l /dev/sda` - то можно увидеть, что его тип – Linux LVM. Про LVM мы ещё поговорим.

```

                                Disk: /dev/sdc
                                Size: 1 GiB, 1073741824 bytes, 2097152 sectors
                                Label: gpt, identifier: 02541524-4E41-2846-AD26-7B9A332308F9

```

Device	Start	End	Sectors	Size	Type
>> /dev/sdc1	2048	411647	409600	200M	Linux filesystem
Free space	411648	2097118	1685471	823M	

Partition UUID: 35832FEC-238D-1149-972C-1CA18C5E4117
 Partition type: Linux filesystem (0FC63DAF-8483-4772-8E79-3D69D8477DE4)

```

[ Delete ] [ Resize ] [ Quit ] [ Type ] [ Help ] [ Write ] [ Dump ]

```

Через командную строку вы также можете воспользоваться псевдографической утилитой `cfdisk` - `sudo cfdisk /dev/sdc`. Также есть и другие утилиты – к примеру `parted`. В конечном счёте, чем пользоваться – решаете вы. Но в целом, практически на всех системах бывают предустановлены `fdisk` и `parted`. На старых системах можно наткнуться на `fdisk`, который не поддерживает GPT, но сейчас это не так актуально. В любом случае, знать много инструментов – хорошо, но зная основы можно разобраться с любой утилитой.

```
[user@centos8 ~]$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                                  8:0      0   20G  0 disk
├─sda1                              8:1      0    1G  0 part /boot
├─sda2                              8:2      0   19G  0 part
│   └─cl_centos8-root              253:0    0   17G  0 lvm  /
│       └─cl_centos8-swap          253:1    0    2G  0 lvm  [SWAP]
└─sdb                               8:16     0    1G  0 disk
    ├─sdb1                         8:17     0   200M  0 part
    ├─sdb4                         8:20     0     1K  0 part
    └─sdb5                         8:21     0   100M  0 part
sdc                                  8:32     0    1G  0 disk
└─sdc1                             8:33     0   200M  0 part
sr0                                 11:0     1  58.2M  0 rom  /run/media/user/VBox_GAs_6.1.14
```

Напоследок, `lsblk` – покажет вам блочные устройства в древовидной форме, где прекрасно видно, что это за диск или раздел и к чему он относится.

```
[user@centos8 ~]$ ll -R /dev/disk/
/dev/disk/:
total 0
drwxr-xr-x. 2 root root 880 Feb 14 18:26 by-id
drwxr-xr-x. 2 root root  60 Feb 14 17:39 by-label
drwxr-xr-x. 2 root root 160 Feb 14 18:26 by-partuuid
drwxr-xr-x. 2 root root 240 Feb 14 18:26 by-path
drwxr-xr-x. 2 root root 120 Feb 14 17:39 by-uuid

/dev/disk/by-id:
total 0
lrwxrwxrwx. 1 root root  9 Feb 14 17:39 ata-VBOX_CD-ROM_VB2-01700376 -> ../../sr0
lrwxrwxrwx. 1 root root 9 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14 -> ../../sdb
lrwxrwxrwx. 1 root root 10 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14-part1 -> ../../sdb1
lrwxrwxrwx. 1 root root 10 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14-part4 -> ../../sdb4
lrwxrwxrwx. 1 root root 10 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14-part5 -> ../../sdb5
lrwxrwxrwx. 1 root root 9 Feb 14 17:39 ata-VBOX_HARDDISK_VB595fd559-360b38c9 -> ../../sda
```

Помните, названия дисков даются `udev`-ом и могут меняться? Однако у дисков и разделов могут быть свои идентификаторы, которые могут помочь вам опознать различные устройства. И это хорошо видно, если запустить команду `ll -R /dev/disk`, где по различным идентификаторам есть символические ссылки, указывающие на текущее название каждого диска и раздела. То есть, как бы не изменилась буква диска – `sda`, `sdb` или `sdc` – идентификатор всегда будет один и тот же, и обращаясь к диску или разделу по идентификатору, вы всегда будете обращаться к нужному диску или разделу.

Получилось довольно много информации, хотя пока мы не затронули тему файловых систем и LVM. Попрактикуйтесь с разделами – создавайте их, удаляйте, используйте различные утилиты – как графические – тот же `Disks` и `Gparted`, так и терминальные – `fdisk`, `cfdisk`, `parted`. Всё это позволит вам лучше закрепить материал и в дальнейшем более уверенно чувствовать себя при работе с дисками и разделами.