

## Изменение размера разделов lvm

Мы с вами прошли продвинутую установку операционной системы Manjaro на жёсткий диск полностью его зашифровав и применив LVM. Возможно многие не совсем понимали для чего потребовалось применять такие меры, ведь установить систему без lvm по продвинутой схеме было бы несколько проще. Соглашусь, проще но не на много, и у нас отпала бы возможность изменять размер зашифрованных разделов внутри основного криптоконтейнера Luks направленного на диск sda. После того, как продвинутую схему поставили человек 20 из первого потока, прошла неделя-другая и мне в личные сообщения стали поступать сообщения, что невозможно обновить систему или установить требуемое программное обеспечение по той причине, что на разделе store-root(корневой раздел) попросту не осталось свободного места. Причём на домашнем разделе store-home ещё есть в запасе 20-40гб. Самое забавное что изменять размер логический разделов предельно просто, с этим действием мы сейчас и ознакомимся.

**Задача:** условно представим что у нас на разделе store-root закончилось свободное место, а на store-home его в излишке. Следовательно вырисовывается задача, отрезать кусок от store-home и данную неразмеченную область свободного пространства "прилепить" к корневому каталогу store-root

Итак, вот список действий которые нам потребуется выполнить:

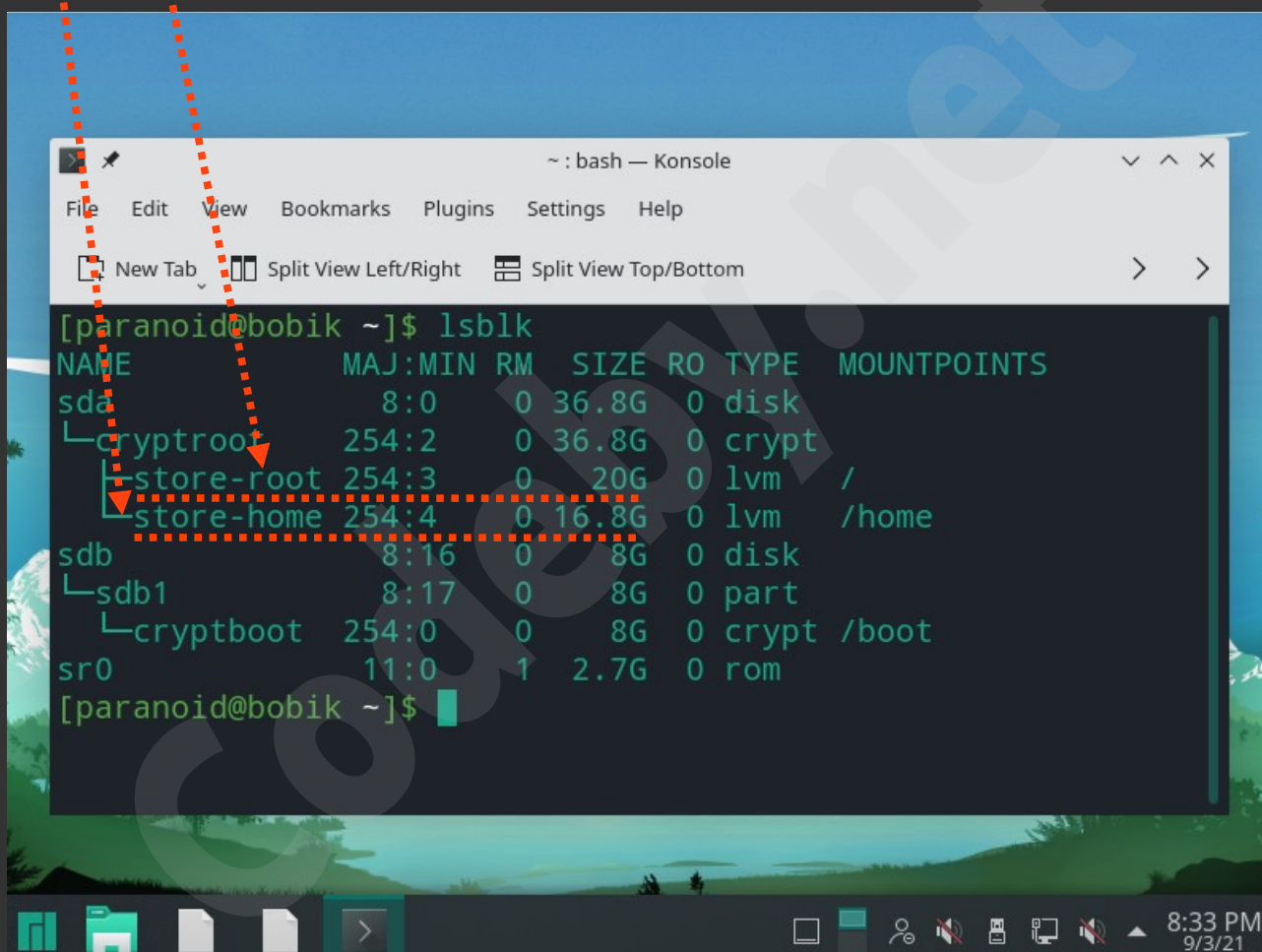
- **Определиться сколько места** мы будем выделять под корневой каталог
- **Перезагрузить компьютер** который настроен по продвинутой схеме и загрузиться на нём с live-образа Manjaro
- **Открыть зашифрованную флешку** `/dev/sdb` и смонтировать её в каталог `/mnt`.
- **После монтирования в папке** `/mnt` появится заголовок и файл ключ, который необходим для разблокировки зашифрованного жёсткого диска `/dev/sda`.
- **Открыть зашифрованный жёсткий диск** применив параметры шифрования которые мы настраивали в процессе установки используя при этом файлы `header.img` и `key.img`
- **Открыть программу KDE Partition Manager** > Отрезать конечное пространство от раздела `store-home` и добавить к разделу `store-root` > применить изменения.

Приступаем: Обратите внимание, у нас есть родительский каталог `cryptroot` и от него идёт ответвление.

**store-root** корневой каталог, который содержит в себе всю операционную систему, её настройки (**20G**).

**store-home** домашний каталог, который содержит в основном незначительные данные, либо личную информацию никак не влияющую на систему в целом (**16.8G**).

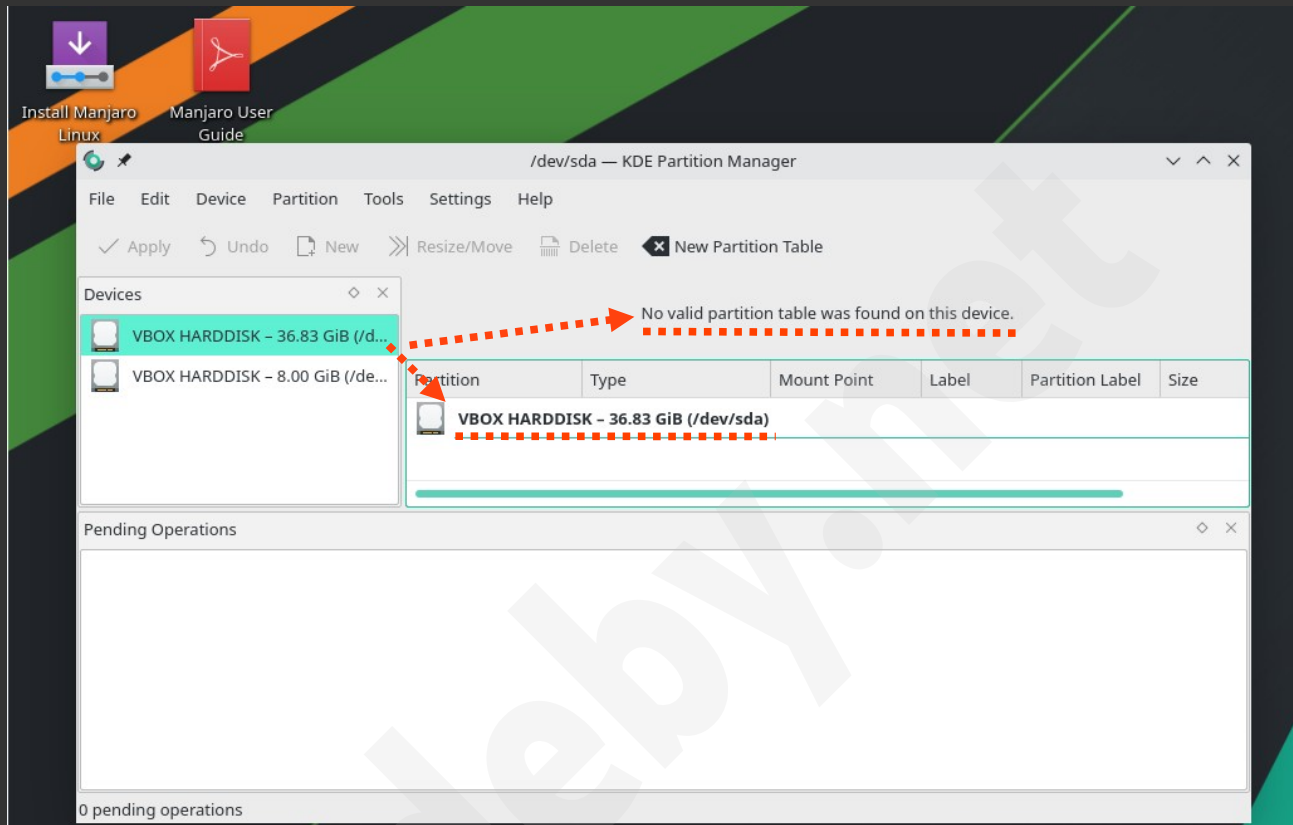
По легенде для работы операционной системы недостаточно выделенных двадцати гигабайт памяти и единственное, что мы можем сделать без полной переустановки и риска потерять все зашифрованные данные - это отрезать часть незадействованной памяти от домашнего каталога.



```
[paranoid@bobik ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
└─ cryptroot 254:2    0 36.8G 0 crypt
   └─ store-root 254:3    0  20G 0 lvm    /
      └─ store-home 254:4    0 16.8G 0 lvm    /home
└─ sdb       8:16    0   8G  0 disk
   └─ sdb1    8:17    0   8G  0 part
      └─ cryptboot 254:0    0   8G  0 crypt /boot
└─ sr0      11:0    1  2.7G 0 rom
```

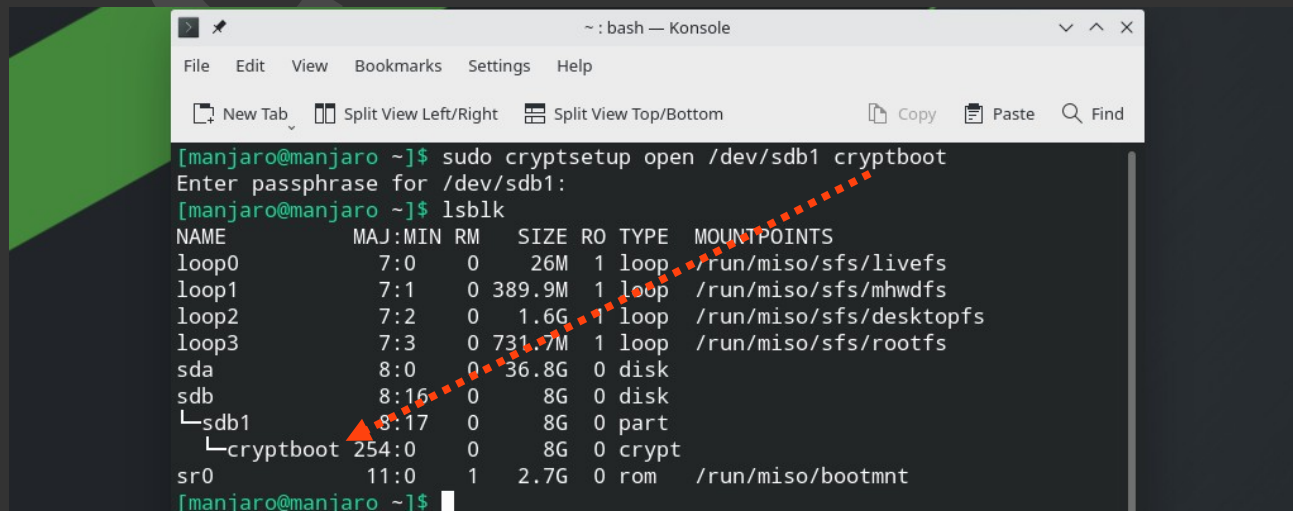
Для нормальной работы системы, обновлений и иных каждодневных развлечений мне будет достаточно добавить к корневому каталогу примерно семь гигабайт. С размерами определились, теперь вставляем в компьютер флешку с live-образом Manjaro либо монтируем в привод `virtualbox`, в зависимости от того как вы ставили систему и перезагружаемся. Без перезагрузки и live образа у вас провести данные мероприятия попросту не выйдет, поскольку разделы состоящие в группе `cryptroot` должны быть отмонтированы, однако их использует система, которая не даст вам этого сделать.

После загрузки live-системы находим в меню программу KDE Partition Manager и открываем её. Обратите внимание, что сейчас раздел **/dev/sda** зашифрован и как следствие не поддаётся изменению, поэтому сначала нам его потребуется открыть ручным способом, по сути сделать часть работ, которые при старте системы автоматически выполняет скрипт `customerscrpthook`. Программа нам вообще пишет, что никакой таблицы разделов на данном накопителе не обнаружено.



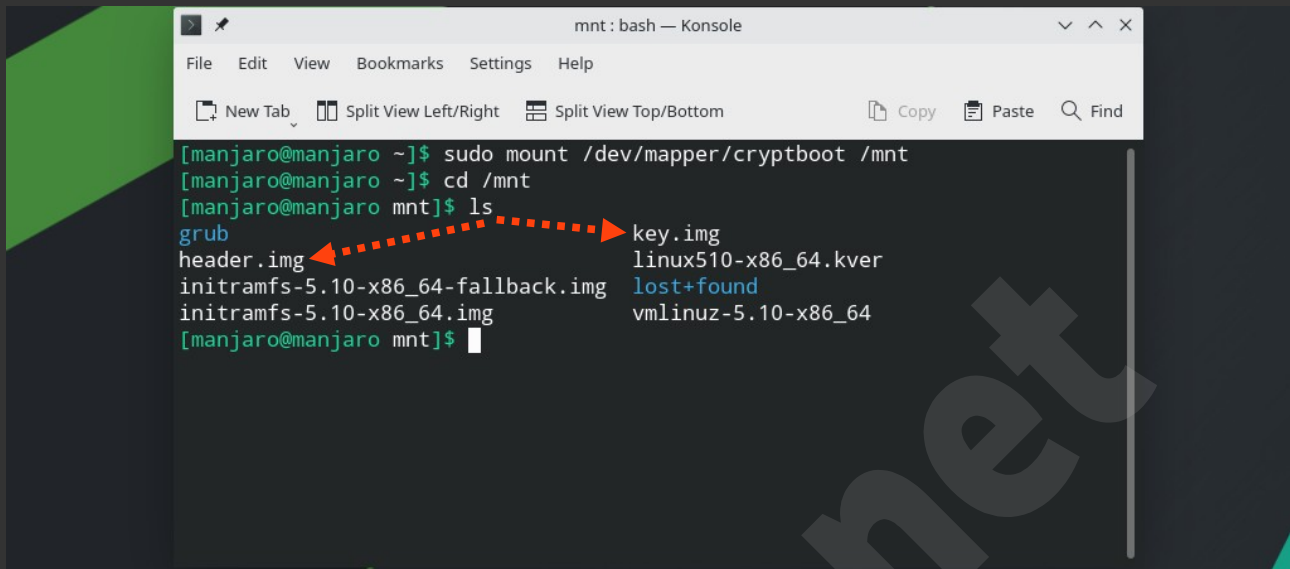
Открываем терминал и выполняем последовательность команд для открытия **sda**:

**# `sudo cryptsetup open /dev/sdb1 cryptboot`** открываем нашу зашифрованную флешку. Если вы производили установку в EFI режиме то открывать вам нужно `/dev/sdb2`. После открытия видим как у нас появился раздел `cryptboot`.



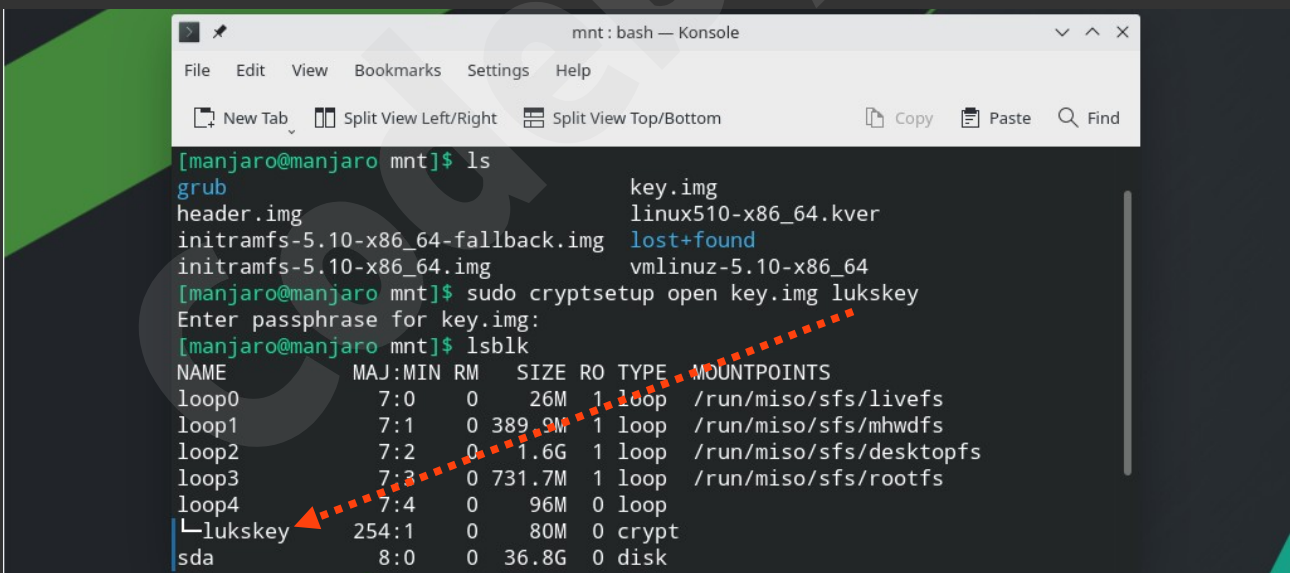
**# sudo mount /dev/mapper/cryptboot /mnt** монтируем содержимое зашифрованной флешки в каталог /mnt

**# cd /mnt** входим в каталог /mnt и можем проверить наличие файлов **header.img** и **key.img**



```
mnt: bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find
[manjaro@manjaro ~]$ sudo mount /dev/mapper/cryptboot /mnt
[manjaro@manjaro ~]$ cd /mnt
[manjaro@manjaro mnt]$ ls
grub                                key.img
header.img                         linux510-x86_64.kver
initramfs-5.10-x86_64-fallback.img lost+found
initramfs-5.10-x86_64.img          vmlinuz-5.10-x86_64
[manjaro@manjaro mnt]$
```

**# sudo cryptsetup open key.img lukskey** открываем зашифрованный файл ключ применив второй пароль! Надеюсь вы помните что первый пароль - это пароль от зашифрованной флешки, который мы вводим при старте системы, а второй (более слабый) от файла ключа. Готово, файл ключ находится в памяти и мы можем открыть с помощью него диск sda.



```
mnt: bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find
[manjaro@manjaro mnt]$ ls
grub                                key.img
header.img                         linux510-x86_64.kver
initramfs-5.10-x86_64-fallback.img lost+found
initramfs-5.10-x86_64.img          vmlinuz-5.10-x86_64
[manjaro@manjaro mnt]$ sudo cryptsetup open key.img lukskey
Enter passphrase for key.img:
[manjaro@manjaro mnt]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0        7:0    0   26M  1 loop /run/miso/sfs/livefs
loop1        7:1    0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2        7:2    0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3        7:3    0 731.7M  1 loop /run/miso/sfs/rootfs
loop4        7:4    0   96M   0 loop
lukskey      254:1   0   80M   0 crypt
sda          8:0    0 36.8G   0 disk
```

**# sudo cryptsetup open --header=/mnt/header.img --key-file=/dev/mapper/lukskey --keyfile-offset=9437 --keyfile-size=8192 /dev/sda**

Открываем жёсткий диск sda применив параметры шифрования, которые мы указывали в самом начале процесса установки по продвинутой схеме. В моём случае смещение по файлу-ключу 9437 байт, а у вас может быть другое значение.

```
mnt: bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro mnt]$ ls
grub                                key.img
header.img                         linux510-x86_64.kver
initramfs-5.10-x86_64-fallback.img lost+found
initramfs-5.10-x86_64.img          vmlinuz-5.10-x86_64
[manjaro@manjaro mnt]$ sudo cryptsetup open --header=/mnt/header.img --key-
file=/dev/mapper/lukskey --keyfile-offset=9437 --keyfile-size=8192 /dev/sda
cryptroot
[manjaro@manjaro mnt]$
```

Диск открыт, у нас появился родительский раздел cryptroot и два дочерних в древе. **store-root** и **store-home**. Извиняюсь за излишнее упрощение, но без этого никуда.

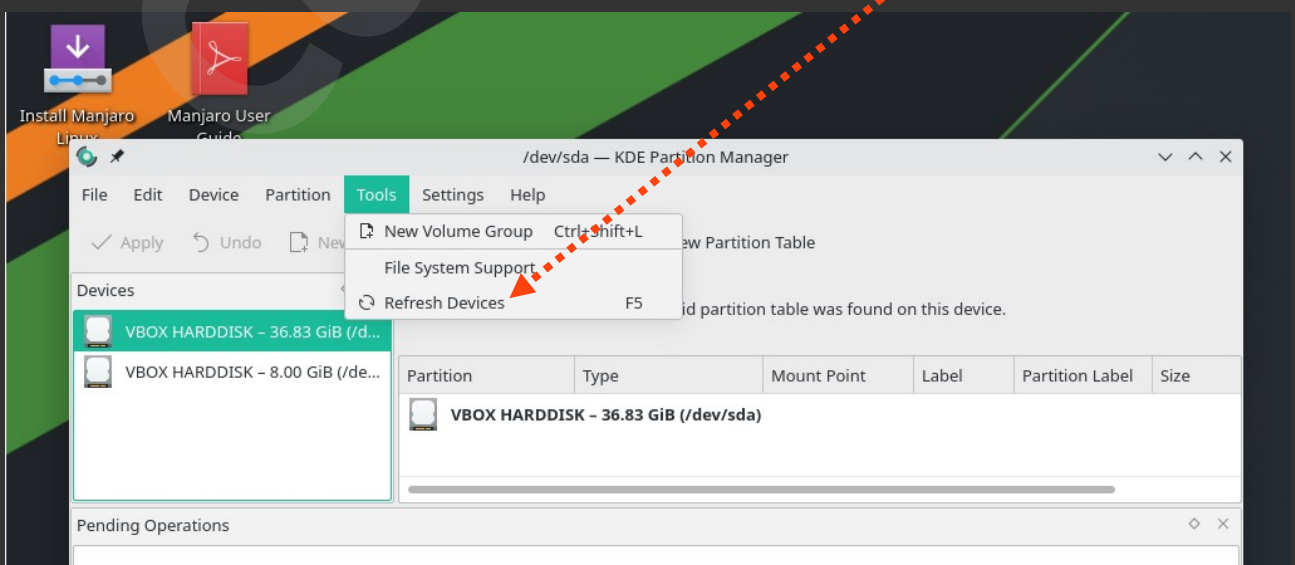
```
mnt: bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
loop0        7:0    0   26M  1 loop /run/miso/sfs/livefs
loop1        7:1    0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2        7:2    0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3        7:3    0 731.7M  1 loop /run/miso/sfs/rootfs
loop4        7:4    0   96M   0 loop
└─lukskey    254:1   0   80M   0 crypt
sda          8:0    0  36.8G  0 disk
└─cryptroot  254:2   0  36.8G  0 crypt
    └─store-root 254:3   0   20G   0 lvm
        └─store-home 254:4   0   16.8G  0 lvm
sdb          8:16   0    8G   0 disk
└─sdb1       8:17   0    8G   0 part
    └─cryptboot 254:0   0    8G   0 crypt /mnt
sr0         11:0    1   2.7G  0 rom  /run/miso/bootmnt

[manjaro@manjaro mnt]$
```

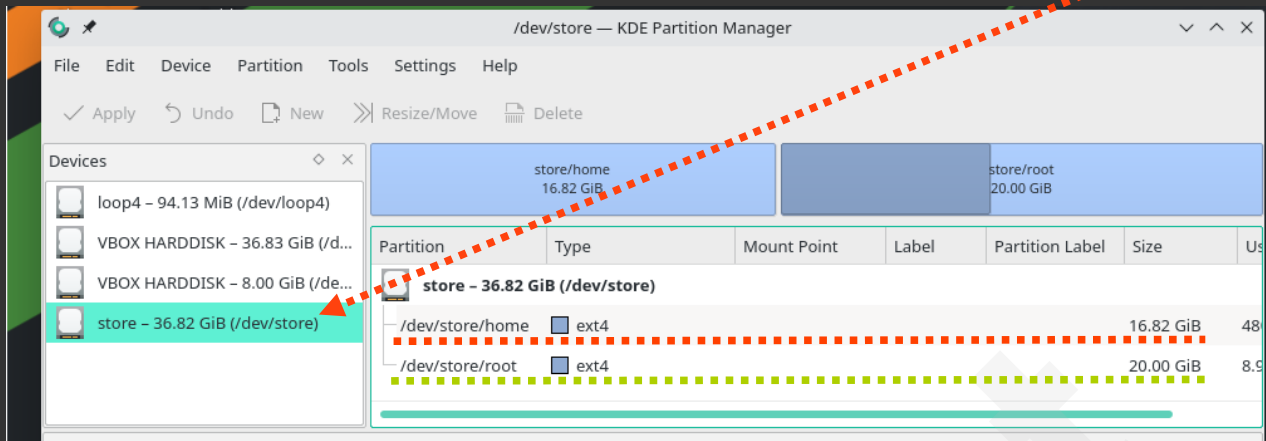
Заметьте, что в отличие от загруженной системы (не live-версия) в live разделы не примонтированы, что очень важно.

Вновь открываем **KDE Partition Manager** > вкладка **"Tools"** > **Refresh devices**



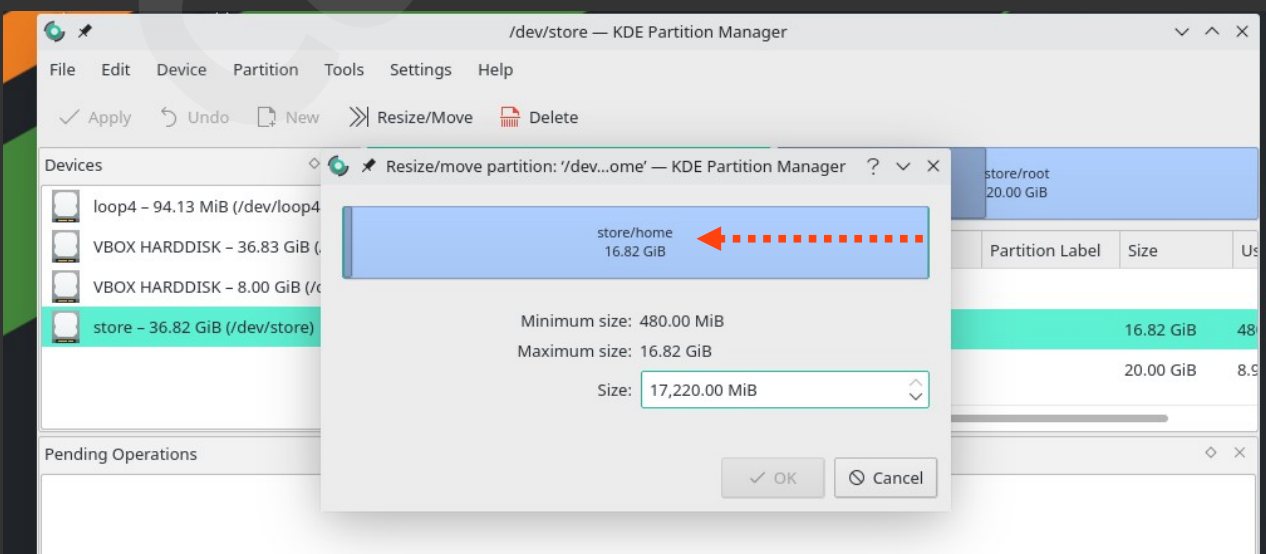
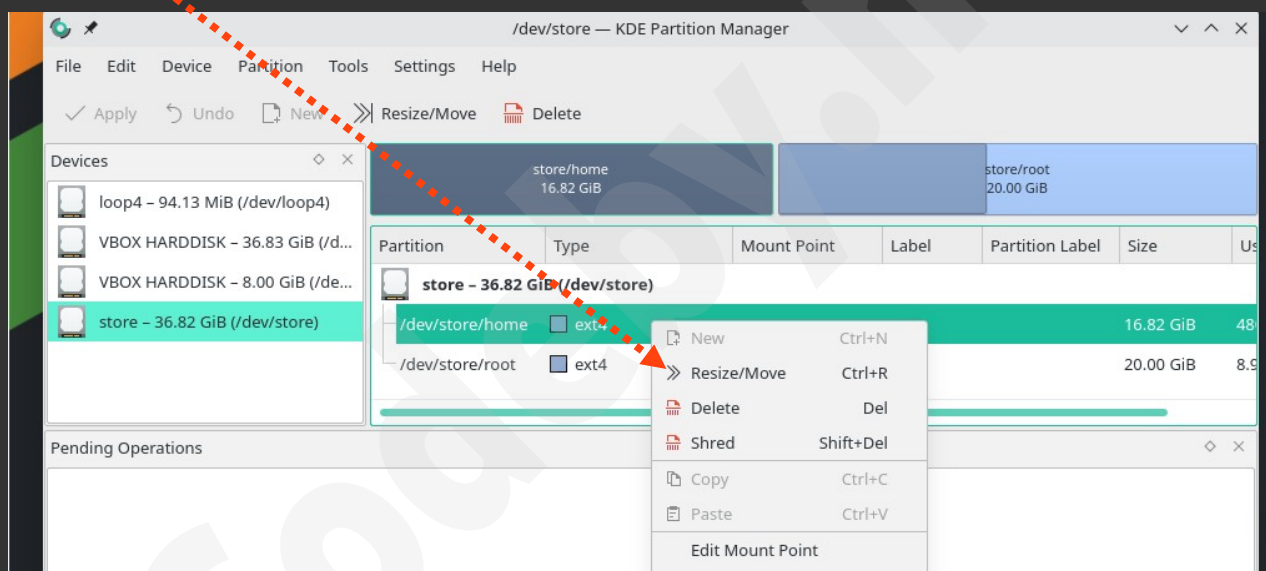


После обновления устройств мы можем наблюдать как появилась логическая группа **store** внутри которой логические разделы **store-home** и **store-root**

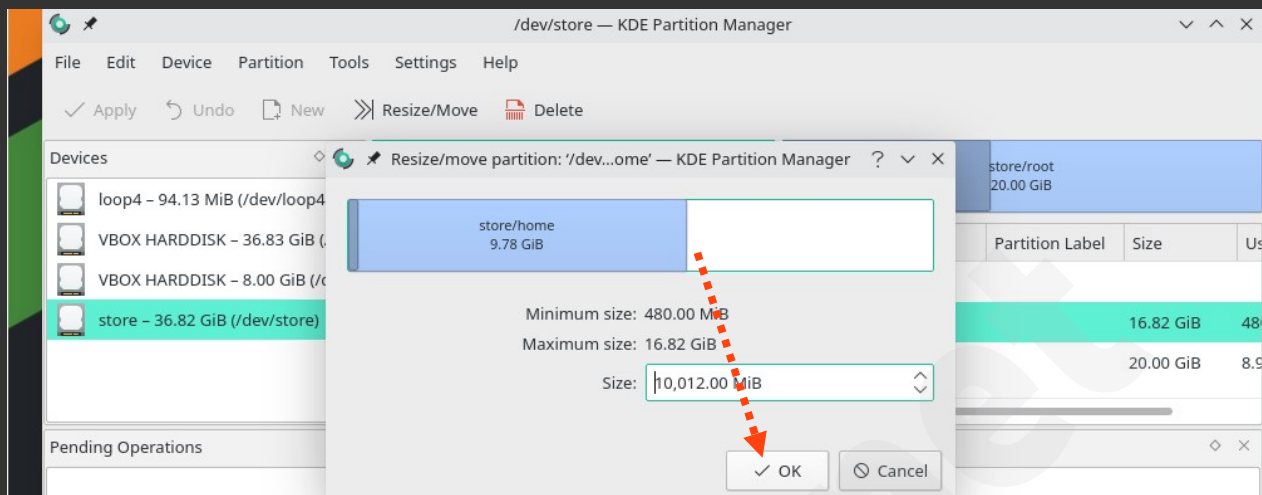


финальная стадия, непосредственно изменение размера логических разделов:

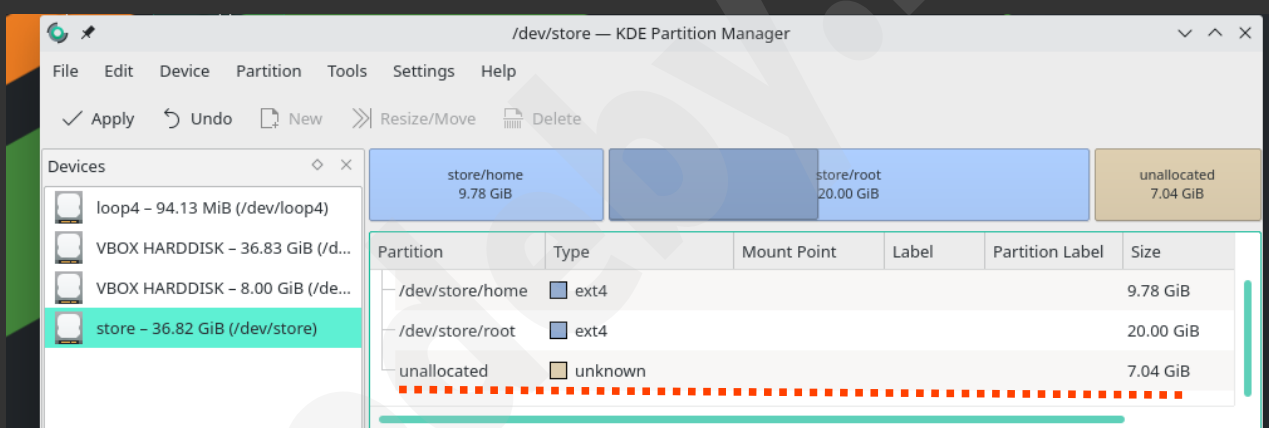
Наводим курсор на логический раздел **/dev/store/home** > жмём ПКМ > выбираем **Resize/Move**



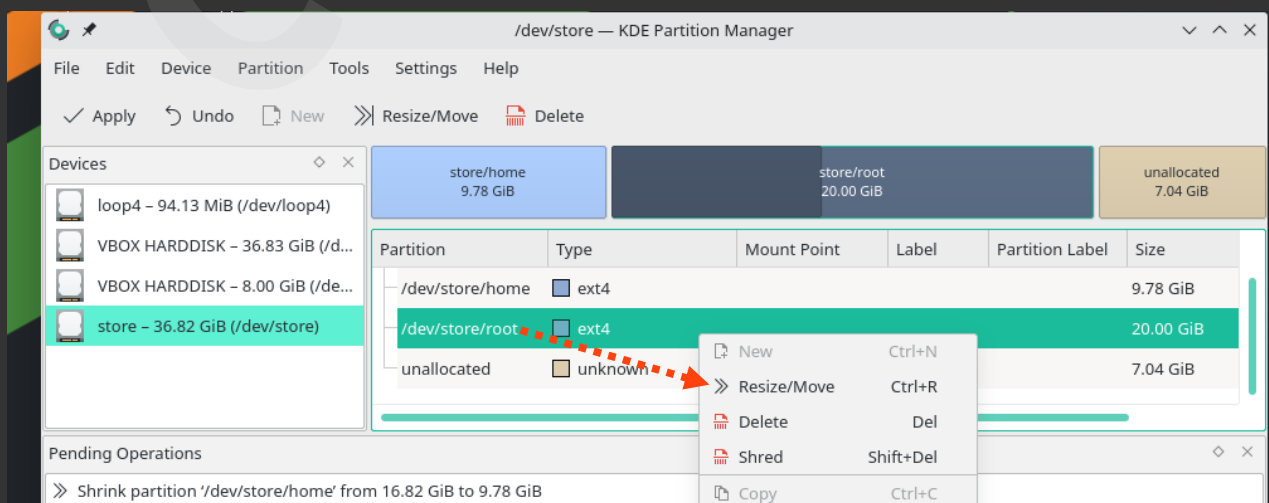
Оттягиваем голубой ползунок до той поры, пока не отрежете необходимое пространство. Отрезать нужно с конца раздела **/dev/store/home**, как указано стрелкой на изображении выше. Изначальный размер раздела был **16.82 Gb** и после моих манипуляций стал **9.78 Gb**. Применяем изменения нажимая кнопку **Ок**



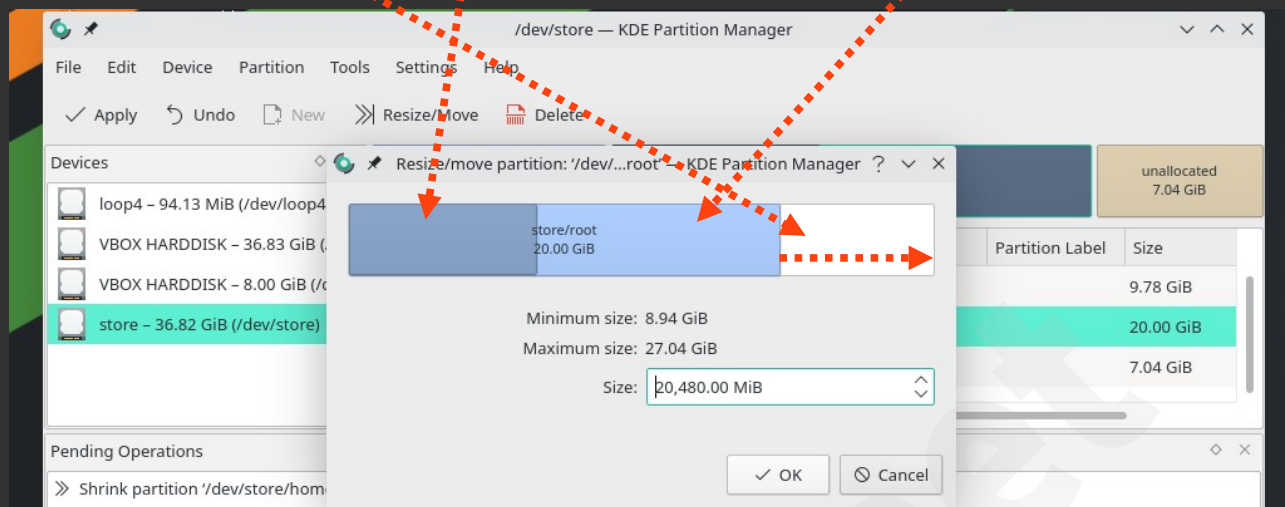
После данной манипуляции появилась неразмеченная область размером 7.04 Gb её то мы и будем добавлять к нашему корневому разделу нуждающемся в расширении.



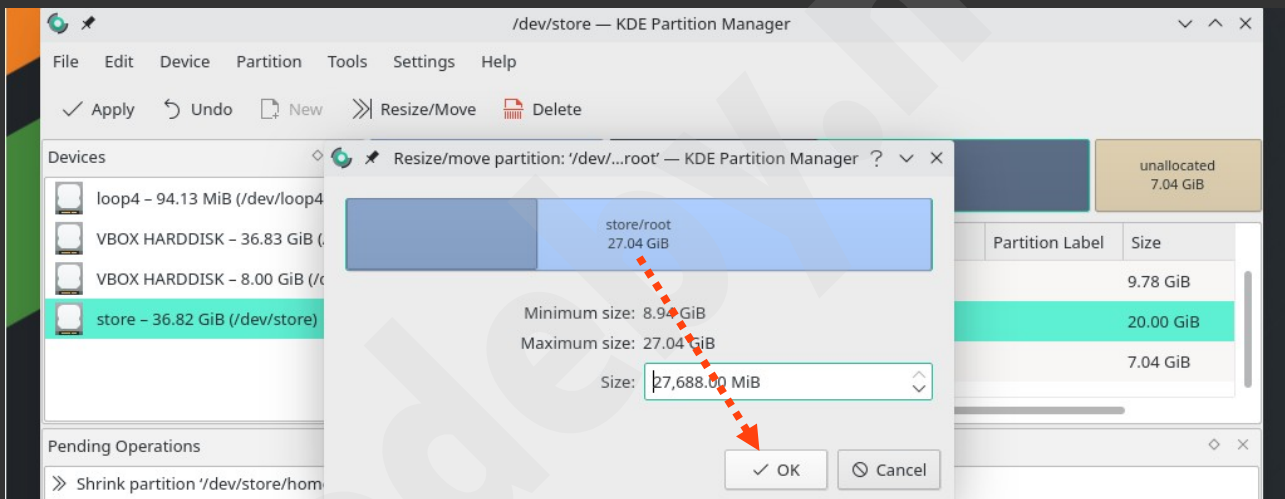
Наводим курсор на логический раздел **/dev/store/root** > жмём ПКМ > выбираем **Resize/Move**



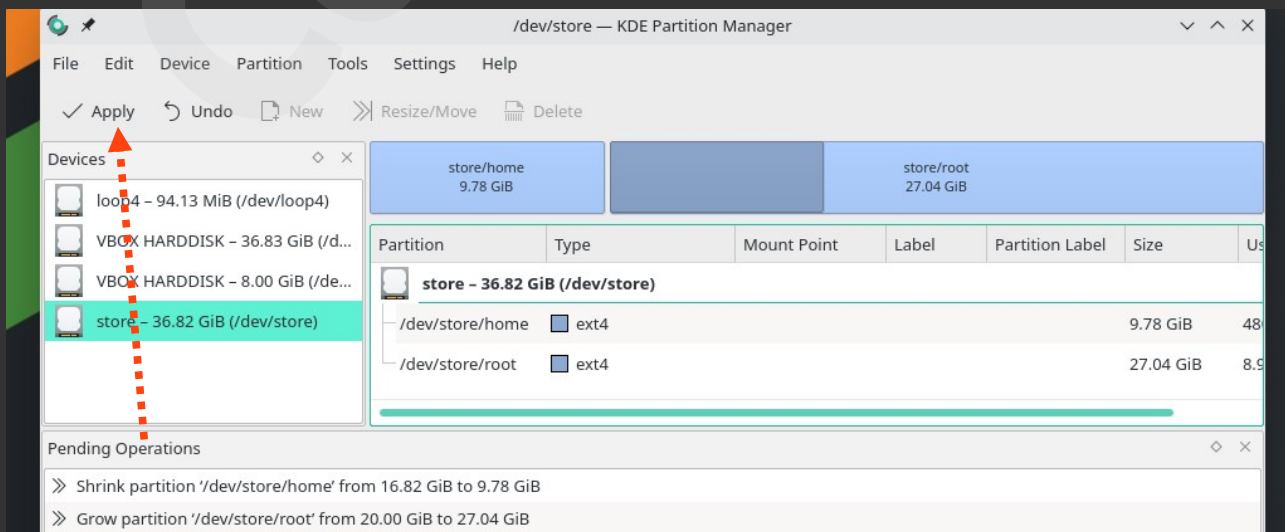
Мы видим ползунок где **тёмно синяя** часть - это занятое пространство, **светло синяя** - свободная и **белая** - это неразмеченная область, которая появилась после предыдущих действий. Двигаем ползунок слева направо занимая голубым цветом всю белую область.



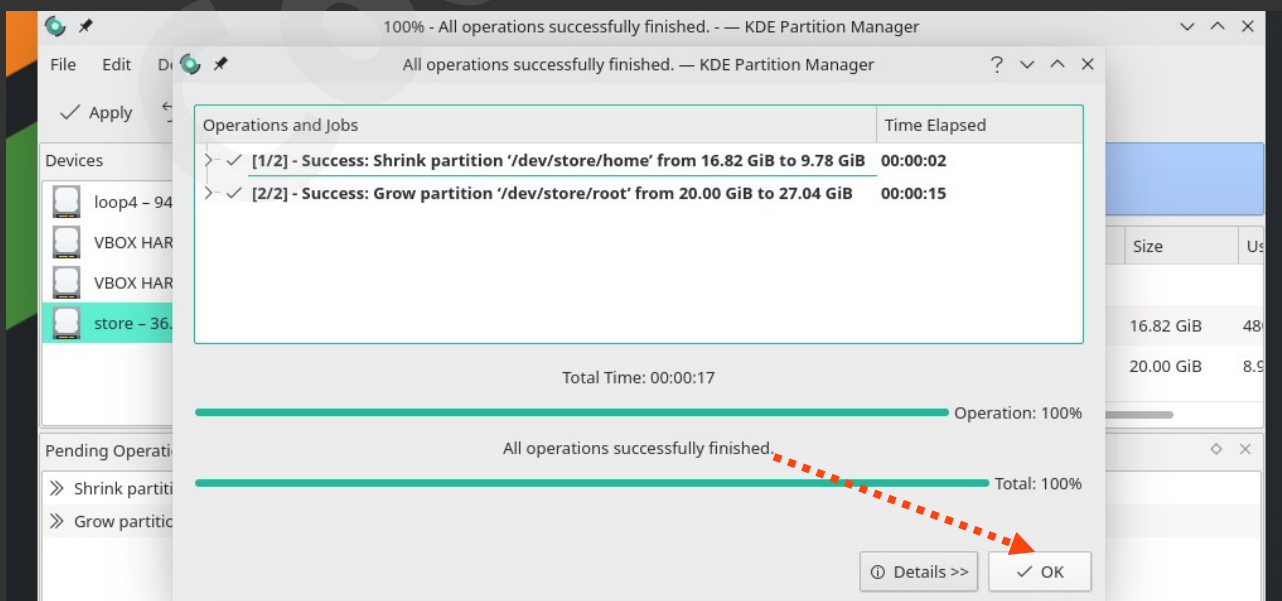
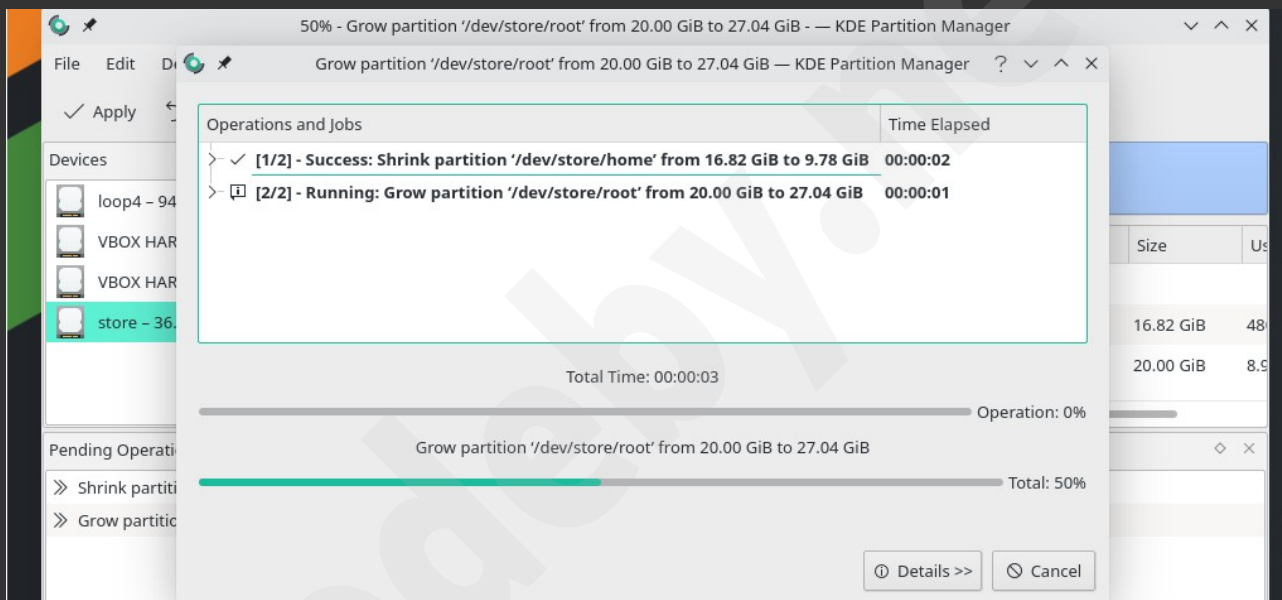
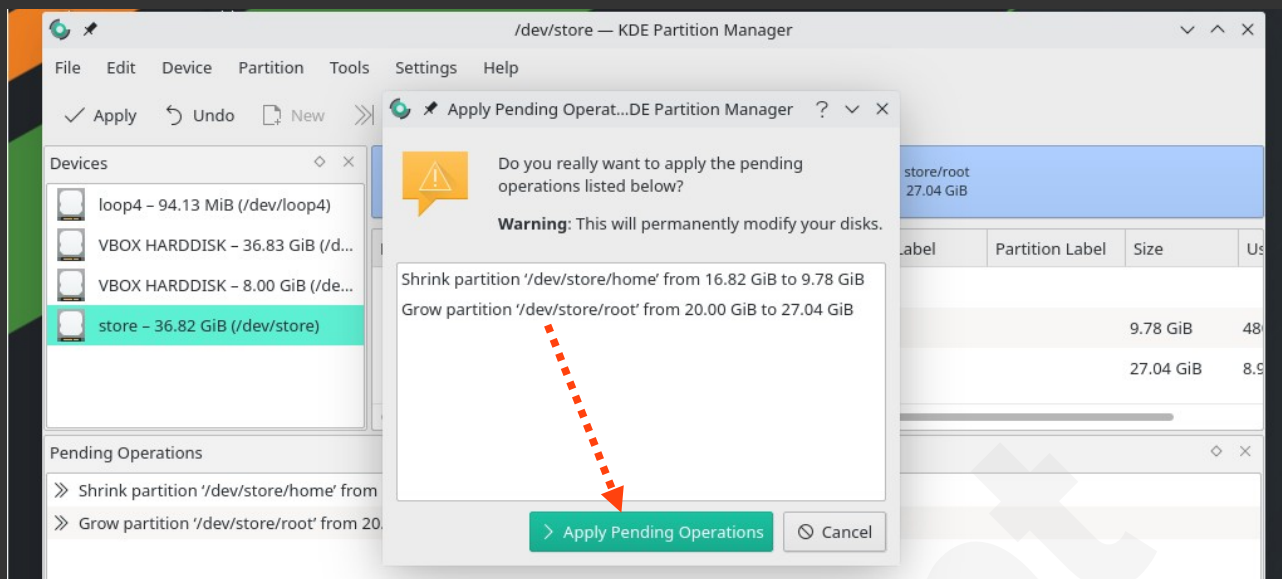
По итогу должно выйти как на изображении ниже > жмём **Ок**



Логический раздел **/dev/store/root** расширен и теперь просто жмём **Apply** для применения изменений.



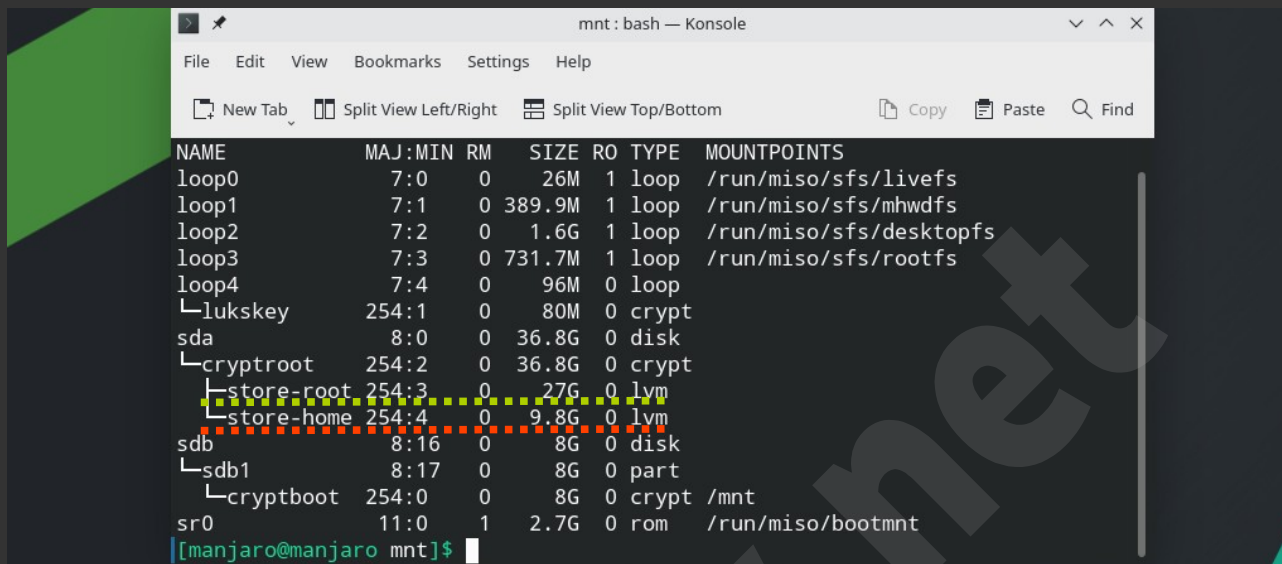




Все запланированные мероприятия выполнены успешно, теперь открываем терминал в live системе и выполняем команду lsblk.

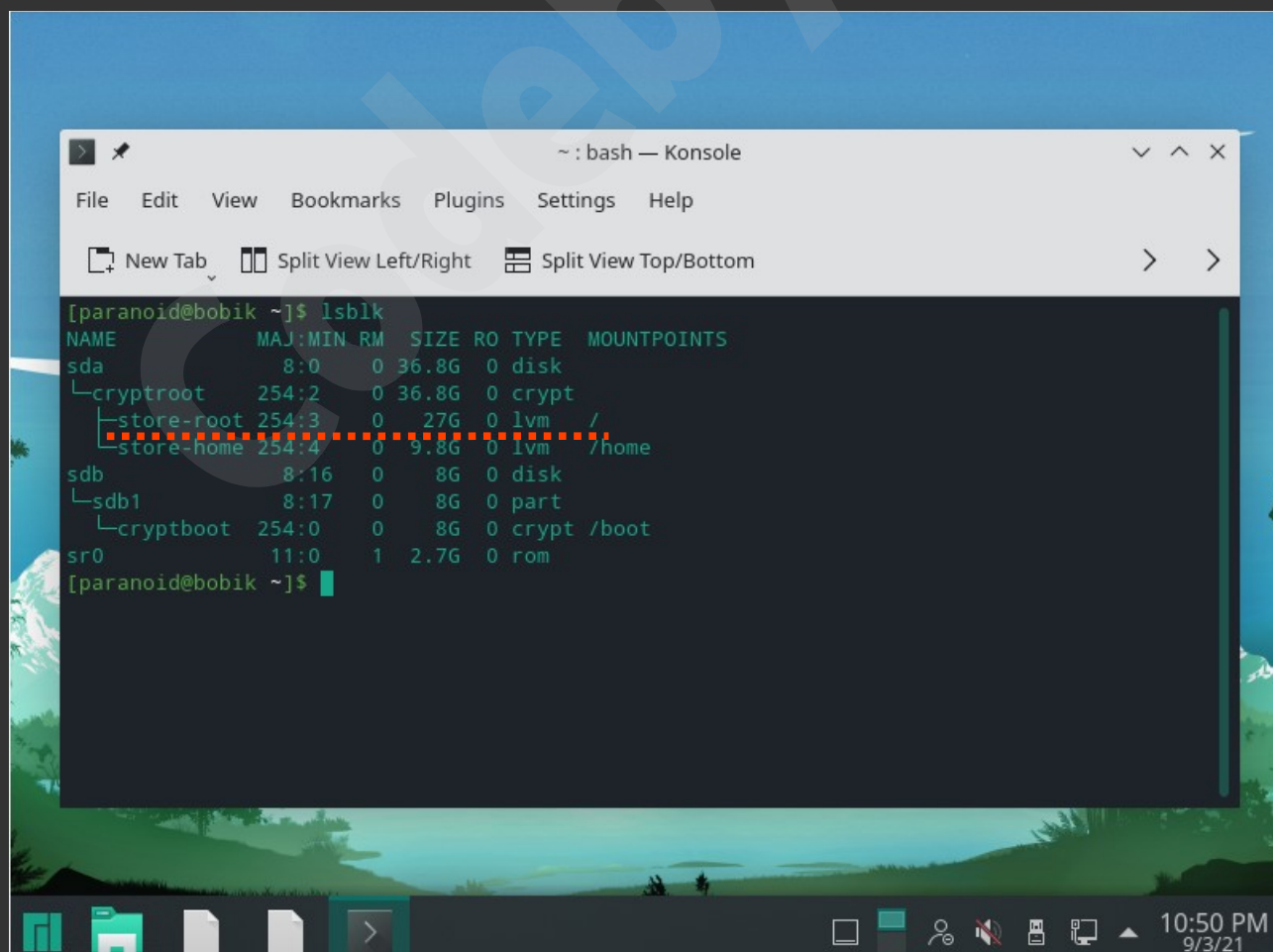
**store-root** теперь имеет размер 27Gb вместо изначальных 20Gb

**store-home** теперь имеет размер 9.8Gb вместо изначальных 16Gb



```
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
loop0        7:0    0   26M  1 loop  /run/miso/sfs/livefs
loop1        7:1    0 389.9M  1 loop  /run/miso/sfs/mhwdfs
loop2        7:2    0   1.6G  1 loop  /run/miso/sfs/desktopfs
loop3        7:3    0 731.7M  1 loop  /run/miso/sfs/rootfs
loop4        7:4    0   96M   0 loop
└─lukskey    254:1    0   80M   0 crypt
sda           8:0    0 36.8G   0 disk
└─cryptroot  254:2    0 36.8G   0 crypt
  └─store-root 254:3    0  27G   0 lvm
  └─store-home 254:4    0  9.8G   0 lvm
sdb           8:16    0    8G   0 disk
└─sdb1       8:17    0    8G   0 part
  └─cryptboot 254:0    0    8G   0 crypt /mnt
sr0          11:0    1    2.7G   0 rom   /run/miso/bootmnt
```

Для полного душевного спокойствия перезагружаемся в нашу систему и проверяем изменения. Как вы имеете честь наблюдать - ничего не сломалось, размер корня расширился и теперь мы можем проходить курс далее.



## Смена паролей в ключевых слотах LUKS (продвинутая схема)

**Внимание!!!** Во избежание потери доступа к зашифрованным данным будьте крайне внимательны, несколько раз перепроверьте и только после этого выполняйте указанные команды. Настоятельно рекомендуется проводить все операции на виртуальной машине `virtualbox` предварительно сохранив слепок состояния (snapshot). В таком случае риск утери данных сведён к нулю.

Все мы помним, что продвинутая схема предполагает использование зашифрованного usb-накопителя в качестве загрузочного звена. Для доступа к содержимому зашифрованного накопителя требуется ввести надёжный пароль с высоким уровнем энтропии. После монтирования флешки в систему у нас появляется зашифрованный файл `key.img` - это ключ для доступа к основному жёсткому диску `sda`. Без знания пароля от ключевого файла расшифровать диск `sda` не представляется возможным, поэтому если его даже и уведёт злоумышленник то получит дырку от бублика. Тем не менее несмотря на всю защищённость и параноидальность схемы присутствует небольшой риск компрометации ключей и паролей доступа. При подобном раскладе существует две основных уязвимости, хотя по большей части мнимые, если соблюдать все правила работы с компьютером. Уязвимости эти следующие.

- Злоумышленник получив физический доступ к вашему компьютеру установил аппаратный кейлоггер, который перехватывает нажатие клавиш и сохраняет в своей внутренней памяти. Далее в зависимости от алгоритма работы устройства, либо сразу передаёт себе полученные сведения по радио/интернет каналу, либо ждёт случая чтобы забрать из вашего железа его в следующий раз. Такое устройство способно перехватить пароль от зашифрованной флешки и ключевого файла `key.img`. Но даже перехватив ваши пароли атакующий не сможет получить никакие данные от диска `SDA` без кражи вашей флешки, поскольку на ней находится основа всего и вся, а именно заголовок `header.img`. Без заголовка расшифровать диск невозможно.
- Злоумышленник перехватил ключ `key.img` в расшифрованном виде при прямом доступе к памяти компьютера, а также сумел перехватить файл `header.img`. В таком случае при последующем запуске компьютера он сможет вручную указать `cryptsetup` использовать перехваченный заголовок и конвертировав ключ из памяти в бинарный файл без каких-то трудностей получит доступ к данным.
- Также возможны разные комбинации из вышеописанного.

Не хочу никого обидеть, но мне трудно представить условия при которых злоумышленник сможет добраться до засекреченных сведений и вот почему:

- ◆ Откуда злоумышленнику знать, что ваш компьютер зашифрован?
- ◆ Откуда злоумышленнику знать, что вы используете LUKS как средство для шифрования? Ведь как мы помним LUKS легко обнаружить, когда заголовок

находится в самом начале зашифрованного диска, но мы то его отделили и положили на флешку.

- ◆ Откуда злоумышленнику знать, что для доступа к компьютеру используется флешка, внутри которой лежит заголовок, зашифрованный файл ключ и ядро?

Без знания этих базовых сведений провести атаку будет предельно трудно. Особенно учитывая что атакующий имеет ограниченный набор информации. В его восприятии вы можете для выхода в сеть использовать флешки с амнезией, внешние жёсткие диски и так далее. Следовательно не зная чёткой схемы очень трудно выстроить вектор атаки.

НО, если атакующий узнает схему и вы для него будете лакомой целью то провести атаку возможно, очень трудно. Тем не менее упускать из внимания этого нельзя. Именно по этой причине создалось это дополнение к основному курсу, а также по просьбе трудящихся)

Соблюдая нижеследующие правила вы сводите на нет любую атаку направленную против вас:

- Не допускайте, чтобы посторонние знали что у вас есть отдельно рабочий и домашний компьютер.
- Никому и никогда старайтесь не рассказывать что у вас зашифрованный компьютер
- Никому и никогда не рассказывайте по какому принципу устроено шифрование, какое программное обеспечение применялось, как именно и в какой очередности происходит загрузка системы.
- Всегда проверяйте ваш компьютер и помещение в котором он находился на предмет постороннего присутствия. Тут все методы хороши, системы мониторинга, простые секретки по типу зубочистки в проём двери и тд.
- Вставьте в рабочий компьютер ещё один диск и выставьте его первым в приоритете загрузки Bios/Efi > Установите на него Windows 10 и создайте видимость, что работа ведётся именно с этой системы, только никакого компромата, так, по мелочи. Не забудьте пароль администратора на вход в систему, нужно же создать видимость хоть какой-то попытки защититься.

Всё же несмотря на вышеописанное менять пароли и ключевые файлы для ключевых разделов нужно уметь, так, на всякий случай. Тем более ничего сложного в этом нет и является совершенно заурядной задачей. Я надеюсь вы помните по первой главе, что одной из особенностей LUKS является восемь ключевых слотов. Если сказать проще, то благодаря этому мы можем при желании один зашифрованный диск открывать разными паролями. Назначать восемь паролей это конечно сильно, и смысла не имеет, однако именно эта особенность позволяет за 5 минут сменить все скопрометированные пароли и ключи. Для начала ознакомимся с основными командами cryptsetup.

**# sudo cryptsetup luksDump /dev/sdb1** данная команда выведет на экран всю информацию о зашифрованной флешке или любом другом разделе в начале которого присутствует заголовок LUKS. На /dev/sda заголовка нет, потому там немного иная история.

**# sudo cryptsetup luksDump /dev/sdb1 | grep Slot** команда выведет на экран терминала информацию о ключевых слотах. Какие из них используются, а какие свободные. Как я и упоминал ранее всего у нас восемь слотов в порядке очереди от 0 до 7. При шифровании базовыми средствами пароль находится в нулевом слоте.

**# sudo cryptsetup luksAddKey /dev/sdb1** данная команда позволяет добавить новый пароль в следующий по очерёдности свободный ключевой слот. Если после этого не удалить старый пароль, то для открытия зашифрованного диска у нас уже будут работать два пароля.

**# cryptsetup luksAddKey /dev/sdb1 -S X** команда полный аналог предыдущий за тем лишь исключением что вместо X мы можем указать в какой из доступных ключей будет записан наш пароль.

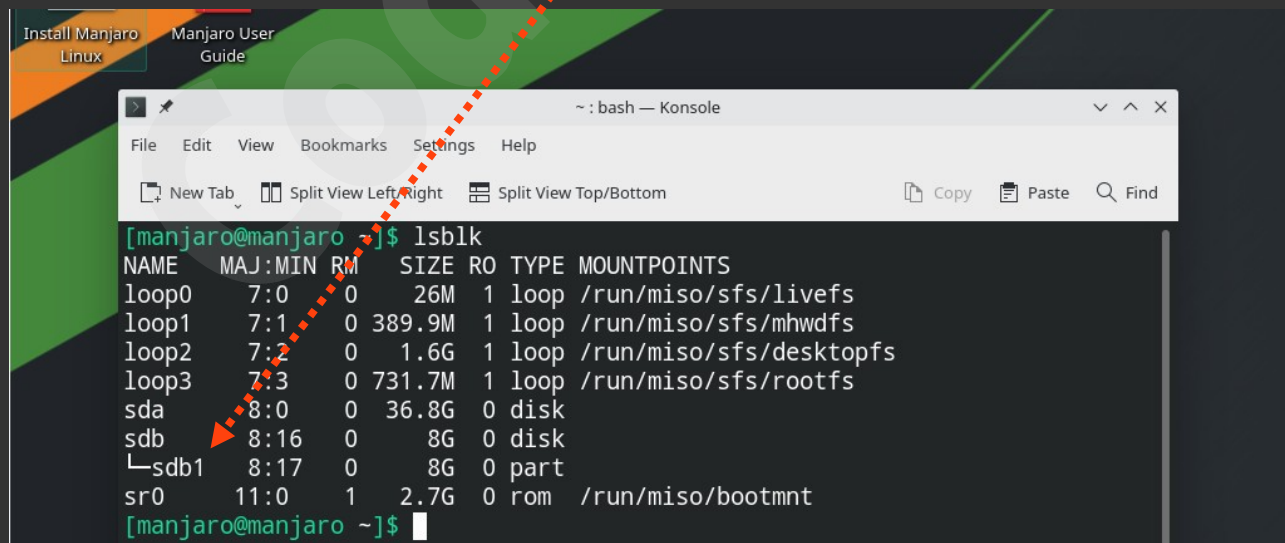
**# cryptsetup luksRemoveKey /dev/sdb1** команда позволяет удалить ключ из слота. Достаточно ввести тот пароль, который мы желаем удалить и слот автоматически освободится. После этого данным паролем диск больше открыть будет невозможно.

**# cryptsetup luksKillSlot /dev/sdb1 1** команда аналог предыдущей за исключением того что позволяет удалять сразу требуемый слот. Особой разницы в применении нет. В данном случае удалится первый слот с ключом.

Итак, базовые команды +- разобрали теперь приступим к практической части.

## Меняем пароль для зашифрованной флешки

**# lsblk** выполняем данную команду чтобы точно узнать название нашей флешки в системе, в моём случае это **/dev/sdb** и раздел **/dev/sdb1** зашифрован, поскольку как вы помните я в Legacy режиме.



```
[manjaro@manjaro ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0        7:0    0   26M  1 loop /run/miso/sfs/livefs
loop1        7:1    0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2        7:2    0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3        7:3    0 731.7M  1 loop /run/miso/sfs/rootfs
sda          8:0    0  36.8G  0 disk
sdb          8:16   0    8G   0 disk
└─sdb1       8:17   0    8G   0 part
sr0         11:0    1   2.7G  0 rom  /run/miso/bootmnt
```



**# sudo cryptsetup luksDump /dev/sdb1 | grep Slot** проверяем доступные слоты LUKS, как видим нулевой занят, а остальные семь свободны и в любой из них мы можем забить новый пароль.

```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro ~]$ sudo cryptsetup luksDump /dev/sdb1 | grep Slot
Key Slot 0: ENABLED
Key Slot 1: DISABLED
Key Slot 2: DISABLED
```

**# sudo cryptsetup luksAddKey /dev/sdb1** добавляем новый пароль в последующий свободный ключевой слот. После ввода команды у нас запросит подтверждение, а именно для добавления нового пароля потребуется ввести старый(предыдущий). Далее вводим два раза новый пароль с высокой степенью энтропии и проверяем ключевые слоты. Обратите внимание, теперь у нас занят второй ключевой слот Slot 1

```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro ~]$ sudo cryptsetup luksAddKey /dev/sdb1
Enter any existing passphrase:
Enter new passphrase for key slot:
Verify passphrase:
[manjaro@manjaro ~]$ sudo cryptsetup luksDump /dev/sdb1 | grep Slot
Key Slot 0: ENABLED
Key Slot 1: ENABLED
Key Slot 2: DISABLED
```

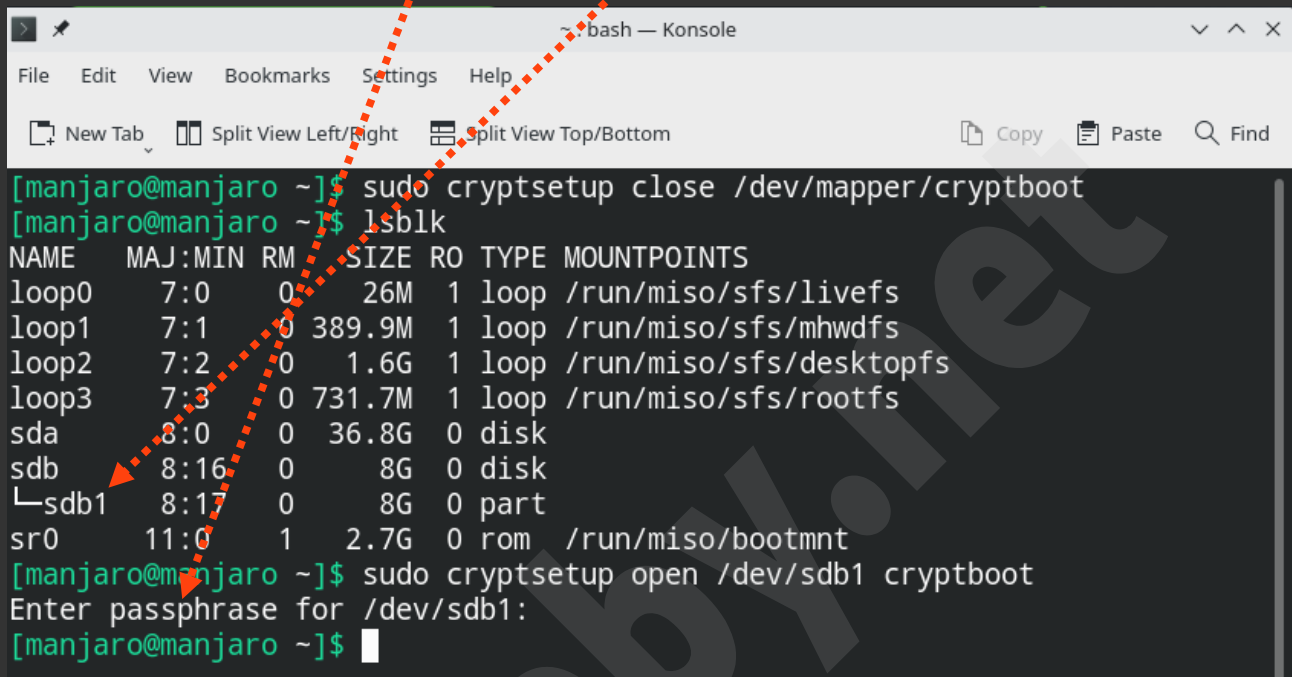
**# sudo cryptsetup open /dev/sdb1 cryptboot** после того как мы добавили новый пароль для первого раздела нашей зашифрованной флешки, мы пытаемся открыть раздел новым паролем из первого ключевого слота. Поскольку я тестирую на виртуальных машинах исключительно для курса, то нет никакой целесообразности использовать надёжные пароли. Исходя из этого старый пароль в ключевом слоте Slot 0 "11111111", а новый пароль в ключевом слоте Slot 2 "22222222". При требовании ввести пароль от раздела вводим 22222222. Раздел успешно открылся, как видно на скриншоте ниже!

```
[manjaro@manjaro ~]$ sudo cryptsetup open /dev/sdb1 cryptboot
Enter passphrase for /dev/sdb1:
[manjaro@manjaro ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0        7:0    0   26M  1 loop /run/miso/sfs/livefs
loop1        7:1    0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2        7:2    0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3        7:3    0 731.7M  1 loop /run/miso/sfs/rootfs
sda          8:0    0 36.8G  0 disk
sdb          8:16   0    8G  0 disk
└─sdb1       8:17   0    8G  0 part
   └─cryptboot 254:0  0    8G  0 crypt
```

Мы уже убедились что новый пароль работает на ура, теперь проверим чтобы и старым паролем была возможность открыть зашифрованный раздел. Для этого нам потребуется сначала вручную закрыть LUKS, а затем вновь открыть.

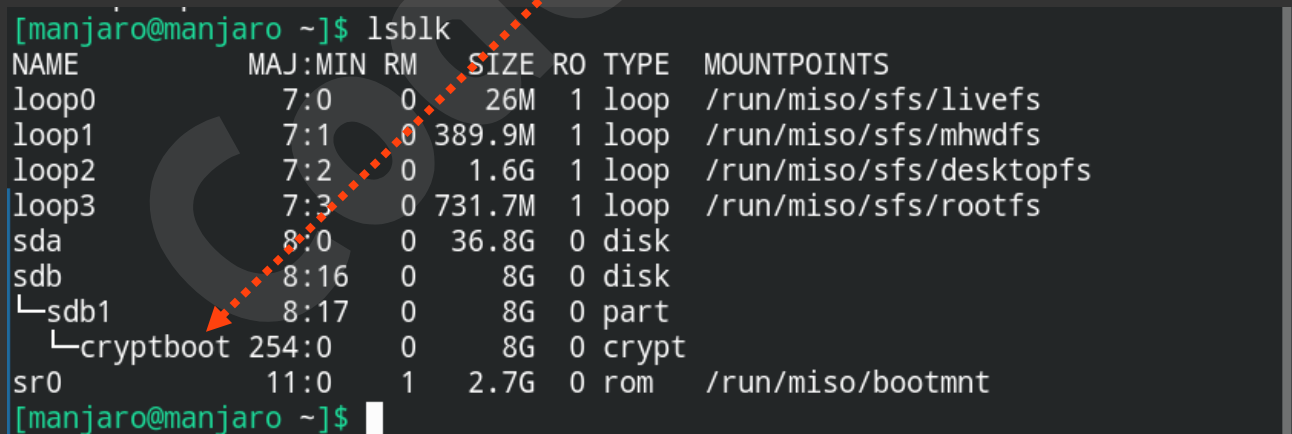
**# sudo cryptsetup close /dev/mapper/cryptroot** закрываем раздел Luks

**# sudo cryptsetup open /dev/sdb1 cryptboot** вновь открываем раздел, но используя для этого старый пароль "11111111"



```
[manjaro@manjaro ~]$ sudo cryptsetup close /dev/mapper/cryptroot
[manjaro@manjaro ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0         7:0    0   26M  1 loop /run/miso/sfs/livefs
loop1         7:1    0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2         7:2    0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3         7:3    0 731.7M  1 loop /run/miso/sfs/rootfs
sda           8:0    0  36.8G  0 disk
sdb           8:16   0    8G    0 disk
└─sdb1        8:17   0    8G    0 part
sr0          11:0    1   2.7G  0 rom  /run/miso/bootmnt
[manjaro@manjaro ~]$ sudo cryptsetup open /dev/sdb1 cryptboot
Enter passphrase for /dev/sdb1:
[manjaro@manjaro ~]$
```

Старым паролем раздел также успешно открылся - это означает, что в настоящее время у нас активно два пароля и мы можем использовать любой из них.



```
[manjaro@manjaro ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0         7:0    0   26M  1 loop /run/miso/sfs/livefs
loop1         7:1    0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2         7:2    0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3         7:3    0 731.7M  1 loop /run/miso/sfs/rootfs
sda           8:0    0  36.8G  0 disk
sdb           8:16   0    8G    0 disk
└─sdb1        8:17   0    8G    0 part
   └─cryptboot 254:0    0    8G    0 crypt
sr0          11:0    1   2.7G  0 rom  /run/miso/bootmnt
[manjaro@manjaro ~]$
```

Согласно нашей легенде старый пароль скомпрометирован и как следствие должен быть заменён, потому что нам никак не подходит текущее положение вещей. После того как мы на сто процентов убедились в работоспособности нового пароля "22222222" - можем приступить к удалению старого пароля.

**# sudo cryptsetup luksRemoveKey /dev/sdb1** после выполнения команды мы должны ввести тот пароль, который должен быть удалён из ключевых слотов вводим "11111111". cryptsetup

автоматически определит что пароль "1111111" соответствует ключевому слоту **Slot 0** и очистит его.

```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro ~]$ sudo cryptsetup luksRemoveKey /dev/sdb1
Enter passphrase to be deleted:
```

**# sudo cryptsetup open /dev/sdb1 cryptboot** пытаемся открыть зашифрованный раздел только что удалённым ключом. Но у нас ничего не получается и мы получаем ошибку **No key available with this passphrase**. В этом нет ничего удивительного, ведь **Slot 0** которому соответствовал наш старый пароль отныне пуст и не задействован.

```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro ~]$ sudo cryptsetup open /dev/sdb1 cryptboot
Enter passphrase for /dev/sdb1:
No key available with this passphrase.
Enter passphrase for /dev/sdb1:
No key available with this passphrase.
Enter passphrase for /dev/sdb1: Error reading passphrase from terminal.
[manjaro@manjaro ~]$ sudo cryptsetup luksDump /dev/sdb1 | grep Slot
Key Slot 0: DISABLED
Key Slot 1: ENABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
[manjaro@manjaro ~]$
```

Мы достигли своей цели. На текущем этапе мы удалили старый скомпрометированный пароль "1111111" от зашифрованного раздела на флешке, а новый пароль "22222222" теперь используется в качестве основного.

```
[manjaro@manjaro ~]$ sudo cryptsetup open /dev/sdb1 cryptboot
Enter passphrase for /dev/sdb1:
[manjaro@manjaro ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0        7:0    0   26M  1 loop /run/miso/sfs/livefs
loop1        7:1    0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2        7:2    0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3        7:3    0 731.7M  1 loop /run/miso/sfs/rootfs
sda           8:0    0  36.8G  0 disk
sdb           8:16   0    8G   0 disk
└─sdb1        8:17   0    8G   0 part
   └─cryptboot 254:0   0    8G   0 crypt
```

**# sudo cryptsetup luksDump /dev/sdb1** проверяем параметры шифрования нашего раздела.

Напоминаю, работает с тем разделом на котором имеется заголовок LUKS, с диском sda данная команда не сработает. Данная команда показывает все необходимые параметры зашифрованного раздела, а именно: версию LUKS, шифр, хэш, длину ключа, задействованные ключевые слоты и так далее.

```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro ~]$ sudo cryptsetup luksDump /dev/sdb1
LUKS header information for /dev/sdb1

Version:                1
Cipher name:            serpent
Cipher mode:            xts-plain64
Hash spec:              whirlpool
Payload offset:         4096
MK bits:                512
MK digest:              69 ea 19 bd f4 14 ea d6 4c e8 f6 49 df b9 20 55 f4 c6 c2 e
5
MK salt:                e4 df 67 04 af d3 b8 e2 bd e5 eb e7 ac 4e 20 6a
                        0a 9c 72 0d 60 22 e9 10 d4 35 31 4c 52 dc 07 e6
MK iterations:          58514
UUID:                  adae9573-d5f8-4159-8c7d-2d3ee700d61b

Key Slot 0: DISABLED
Key Slot 1: ENABLED
      Iterations:      1548856
      Salt:            08 19 6c 9a 46 a7 ce 34 64 42 09 74 83 e9
67 11
```

### Изменения пароля для key.img

Мы поменяли основной пароль для зашифрованного usb-накопителя. Теперь же наша задача изменить пароль на открытие ключевого файла, размер которого 100мб. Для этого мы открываем зашифрованный раздел на флешке, монтируем его в каталог /mnt.

**# sudo mount /dev/mapper/cryptboot /mnt**

**# ls /mnt** мы видим файл ключ и заголовок.

```
[manjaro@manjaro ~]$ sudo mount /dev/mapper/cryptboot /mnt
[manjaro@manjaro ~]$ ls /mnt
grub                                key.img
header.img                         linux510-x86_64.kver
initramfs-5.10-x86_64-fallback.img lost+found
initramfs-5.10-x86_64.img          vmlinuz-5.10-x86_64
[manjaro@manjaro ~]$
```

**# sudo cryptsetup luksAddKey key.img** добавляем новый пароль в свободный ключевой слот  
**key.img**

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro mnt]$ ls
grub          initramfs-5.10-x86_64-fallback.img  key.img          lost+found
header.img    initramfs-5.10-x86_64.img           linux510-x86_64.kver  vmlinuz-5.10-x86_64
[manjaro@manjaro mnt]$ sudo cryptsetup luksAddKey key.img
Enter any existing passphrase:
Enter new passphrase for key slot:
Verify passphrase:
[manjaro@manjaro mnt]$
```

**# sudo cryptsetup open key.img lukskey** открываем зашифрованный ключ новым паролем, как видим всё открывается должным образом.

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro mnt]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
loop0         7:0    0   26M  1 loop  /run/miso/sfs/livefs
loop1         7:1    0 389.9M  1 loop  /run/miso/sfs/mhwdfs
loop2         7:2    0   1.6G  1 loop  /run/miso/sfs/desktopfs
loop3         7:3    0 731.7M  1 loop  /run/miso/sfs/rootfs
sda           8:0    0  36.8G  0 disk
sdb           8:16   0    8G    0 disk
└─sdb1        8:17   0    8G    0 part
   └─cryptboot 254:0   0    8G    0 crypt /mnt
sr0          11:0    1   2.7G  0 rom   /run/miso/bootmnt
[manjaro@manjaro mnt]$ sudo cryptsetup open key.img lukskey
Enter passphrase for key.img:
[manjaro@manjaro mnt]$
```

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

Enter passphrase for key.img:
[manjaro@manjaro mnt]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
loop0         7:0    0   26M  1 loop  /run/miso/sfs/livefs
loop1         7:1    0 389.9M  1 loop  /run/miso/sfs/mhwdfs
loop2         7:2    0   1.6G  1 loop  /run/miso/sfs/desktopfs
loop3         7:3    0 731.7M  1 loop  /run/miso/sfs/rootfs
loop4         7:4    0    96M  0 loop
└─lukskey    254:1   0    80M  0 crypt
sda           8:0    0  36.8G  0 disk
sdb           8:16   0    8G    0 disk
└─sdb1        8:17   0    8G    0 part
   └─cryptboot 254:0   0    8G    0 crypt /mnt
sr0          11:0    1   2.7G  0 rom   /run/miso/bootmnt
```

**# sudo cryptsetup close lukskey** закрываем файл ключ

**# sudo cryptsetup open key.img lukskey** открываем файл ключ старым паролем. Как вы видите что новый пароль, что старый открывают зашифрованный файл ключ. Следовательно наша задача удалить ключевой слот старого пароля, оставив лишь новый.



```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro mnt]$ sudo cryptsetup close lukskey
[manjaro@manjaro mnt]$ sudo cryptsetup open key.img lukskey
Enter passphrase for key.img:
[manjaro@manjaro mnt]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
loop0         7:0    0   26M  1 loop  /run/miso/sfs/livefs
loop1         7:1    0 389.9M  1 loop  /run/miso/sfs/mhwdfs
loop2         7:2    0   1.6G  1 loop  /run/miso/sfs/desktopfs
loop3         7:3    0 731.7M  1 loop  /run/miso/sfs/rootfs
loop4         7:4    0    96M  0 loop
└─lukskey     254:1    0    80M  0 crypt
```

# **sudo cryptsetup luksDump key.img** проверяем файл ключ на доступность паролей в ключевых слотах, в данное время их должно быть два.

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[manjaro@manjaro mnt]$ sudo cryptsetup luksDump key.img
LUKS header information
Version:          2
Epoch:           4
Metadata area:    16384 [bytes]
Keyslots area:    16744448 [bytes]
UUID:             a4e00b29-065e-491a-956a-1665d923a6de
Label:            (no label)
Subsystem:        (no subsystem)
Flags:            (no flags)

Data segments:
```

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

Keyslots:
0: luks2
   Key:          512 bits
   Priority:      normal
   Cipher:        twofish-xts-plain64
   Cipher key:    512 bits
   PBKDF:         argon2i
   Time cost:     4
   Memory:        698219
   Threads:       2
```

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

1: luks2
   Key:          512 bits
   Priority:      normal
   Cipher:        twofish-xts-plain64
   Cipher key:    512 bits
   PBKDF:         argon2i
   Time cost:     4
   Memory:        726615
   Threads:       2
   Salt:          7a a1 76 46 23 29 87 9a 4c 51 c2 51 6f c7 0b 67
```

После того как мы убедились что у нас два ключевых слота доступно, а зашифрованный раздел открывается двумя паролями - приступаем к последней стадии. Удаляем старый пароль, который находился в нулевом ключевом слоте.

**# sudo cryptsetup luksRemoveKey key.img** вводим старый пароль при соответствующем запросе и тем самым автоматически удаляем слот в котором он находился. При желании можем проверить изменения командой `luksDump`.

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find
[manjaro@manjaro mnt]$ sudo cryptsetup luksRemoveKey key.img
Enter passphrase to be deleted:
[manjaro@manjaro mnt]$ sudo cryptsetup luksDump key.img
LUKS header information
Version:      2
Epoch:      5
```

Теперь доступен только один пароль.

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find
Data segments:
0: crypt
  offset: 16777216 [bytes]
  length: (whole device)
  cipher: twofish-xts-plain64
  sector: 512 [bytes]
Keyslots:
1: luks2
  Key:        512 bits
  Priority:    normal
  Cipher:     twofish-xts-plain64
  Cipher key: 512 bits
  PBKDF:      argon2i
```

Приступаем к окончательной проверке, сначала закрываем открытый ключ `lukskey`, а затем пытаемся его открыть старым паролем, при правильных действиях он больше не должен открыться. Теперь активен только новый пароль.

**# sudo cryptsetup close lukskey**

**# sudo cryptsetup open key.img lukskey**

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find
[manjaro@manjaro mnt]$ sudo cryptsetup close lukskey
[manjaro@manjaro mnt]$ sudo cryptsetup open key.img lukskey
Enter passphrase for key.img:
No key available with this passphrase.
Enter passphrase for key.img:
[manjaro@manjaro mnt]$
```

С применением нового пароля всё открывается как и должно.

```
mnt : bash — Konsole
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find
Enter passphrase for key.img:
[manjaro@manjaro mnt]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
loop0        7:0      0   26M  1 loop /run/miso/sfs/livefs
loop1        7:1      0 389.9M  1 loop /run/miso/sfs/mhwdfs
loop2        7:2      0   1.6G  1 loop /run/miso/sfs/desktopfs
loop3        7:3      0 731.7M  1 loop /run/miso/sfs/rootfs
loop4        7:4      0   96M   0 loop
└─lukskey    254:1     0   80M   0 crypt
sda           8:0      0 36.8G   0 disk
sdb           8:16     0    8G   0 disk
└─sdb1       8:17     0    8G   0 part
   └─cryptboot 254:0     0    8G   0 crypt /mnt
sr0          11:0     1   2.7G   0 rom   /run/miso/bootmnt
[manjaro@manjaro mnt]$
```

Вот мы и проделали описанные в данной главе руководства. Откровенно говоря добраться к вашим паролям атакующему будет непросто, вероятность добраться к вашей системе настолько ничтожна, что можно не заморачиваться. Ежели она всё таки будет, то вы просто будете более осмотрительным и осторожным. Как следствие никакой злополучный ирод или псина немывая не сможет получить доступ без вашего желания.