

Резервная копия загрузочной флешки

На предыдущем шаге мы с вами зашифровали систему таким образом, что заголовок **"header.img"** от жёсткого диска и файл ключ **"key.img"** находятся на внешнем зашифрованном устройстве. Поэтому флешка является единственным звеном с помощью которого можно расшифровать корень **/dev/sda** и загрузить операционную систему. Однако что произойдёт, если по каким-либо причинам вы потеряете доступ к заголовку и ключу? Например usb-накопитель выйдет из строя, вы в спешке уничтожите флешку когда настанет час X, а перед этим не удосужитесь сделать резервную копию и так далее. Ответ предельно прост и ироничен - вы больше никогда не расшифруете систему и не получите сведений внутри зашифрованного диска sda. У меня недавно был забавный случай касающийся входа в операционную систему по продвинутой схеме установки. Дело было вечером, сижу у компьютера как вдруг отключилось электричество тут стоит оговориться, что компьютер порой не выключается по 15 суток в связи с работой в трудных климатических условиях. Нужно это для того, чтобы постоянно поддерживалась рабочая температура внутри корпуса и не происходило снижение оной в отрицательные значения. Так вот компьютер выключился и тут я осознал, что забыл свои пароли для входа в систему, к слову по 20 и 14 символов. В истерических попытках восстановить доступ я терзался 20 минут, пока наконец сработала память рук и я сам того не понимая нашёл правильную комбинацию. Почему это произошло? Во первых причина в редких вводах этих паролей, а поскольку они трудные то и забываются быстро. Такого не происходит если компьютер включается и выключается каждый день, тогда вы его настолько чётко вбиваете в сознание и забиваете в память рук, что руки сами набирают пароль. Во вторых я поленился сделать запись двух загрузочных паролей на лист бумаги и спрятать в укромном месте. Следовательно каков был бы итог не вспомни я эти пароли? Доступ к файловой системе на sda был бы утерян навечно, а вместе с ним и все материалы по курсу **Paranoid**, начиная от видео роликов и заканчивая текстовыми черновиками.

Чтобы не допустить глупую оплошность о которую я чуть сам не споткнулся - вам потребуется сделать следующее:

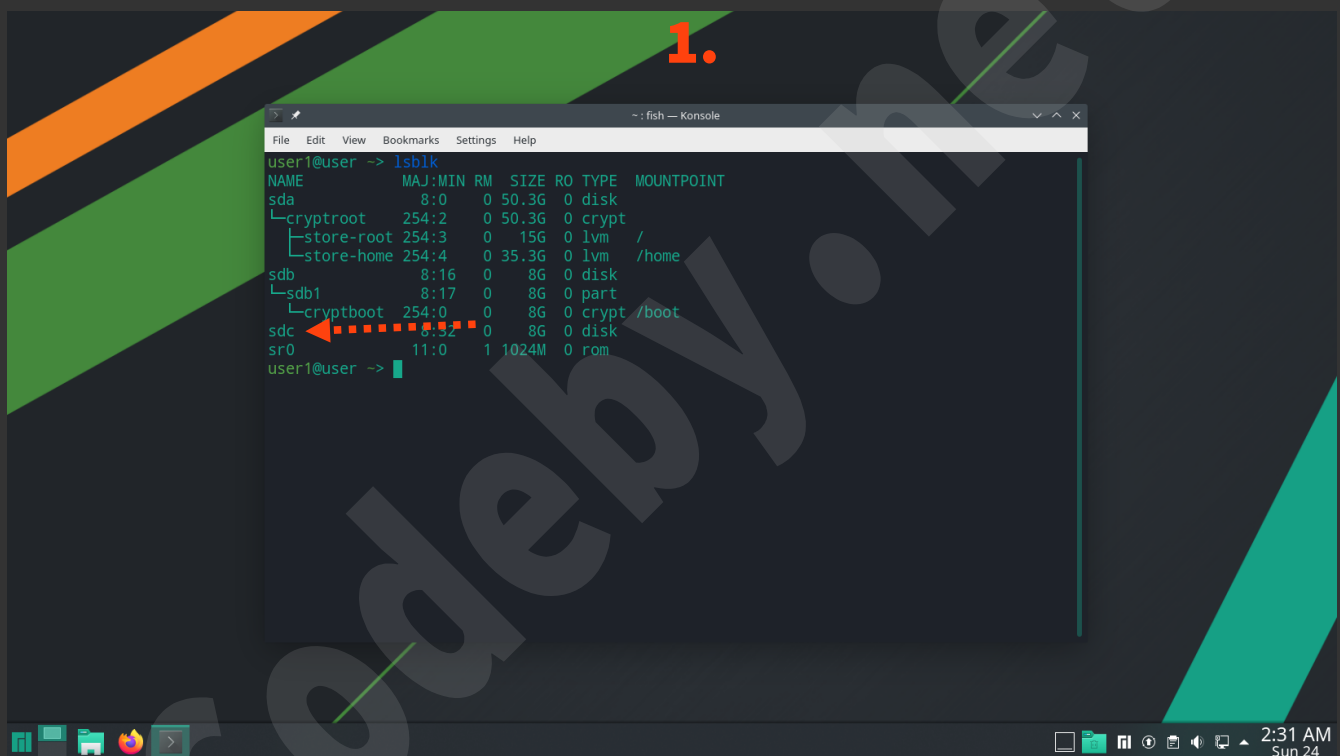
- ✓ Сделать резервную копию **"header.img"**
- ✓ Сделать резервную копию **"key.img"**
- ✓ Сохранить в надёжном месте пароли от **/boot** раздела шифрофлешки и ключа **key.img**, желательно на физическом носителе. На тот случай, если пароли вылетят из головы.

Сделать бэкап заголовка и файла ключа предельно просто из работающей системы, достаточно зайти в смонтированную папку **/boot** и скопировать их в надёжное место. Однако в дальнейшем вы всё равно рано или поздно столкнётесь с потребностью создать новую загрузочную флешку. Этим мы сегодня и займёмся, чтобы при возникновении такой потребности вы знали что делать.

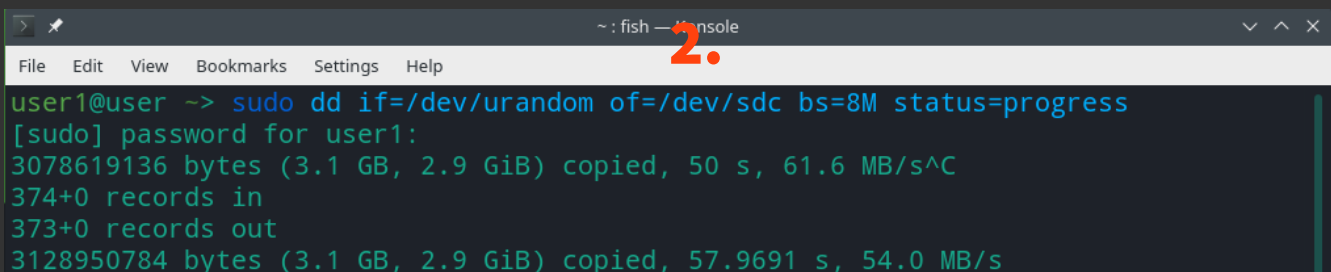
Прежде чем приступить, для более простого понимания и восприятия дальнейших действий приведу краткий список требуемых задач:

- **Загружаем настроенную операционную** систему по продвинутой схеме.
- **Создаём новую флешку** для загрузки системы:

- Подключаем новую флешку к компьютеру и форматируем её
- Создаём на флешке раздел для хранения зашифрованных данных **/dev/sdX1**. Где **X** выданная буква нашей флешке
- Шифруем только что созданный раздел с помощью **LUKS1**
- Открываем новое зашифрованное устройство и присваиваем ему имя **cryptbootbackup**, а также форматируем его в файловую систему **ext2**.
- Монтируем флешку в раздел **/mnt**
- Копируем в смонтированную файловую систему на зашифрованной флешке заголовок и файл-ключ.
- **Перезагружаемся в live-образ Manjaro:**
- Открываем зашифрованный жёсткий диск **sda** используя файл-ключ и заголовок
- Монтируем все устройства и разделы в папку **/mnt**.
- Входим в **chroot**, устанавливаем ядро и настраиваем конфигурационные файлы системы.



sudo dd if=/dev/urandom of=/dev/sdc bs=8M status=progress забиваем **sdc** случайными данными.



sudo cfdisk /dev/sdc приступаем к созданию таблицы разделов на **/dev/sdc**

```
~ : fish — Konsole
File Edit View Bookmarks Settings Help
user1@user -> sudo cfdisk /dev/sdc
```

Тип разметки обязательно выбираем **dos**, в противном случае данная флешка не сможет выступать загрузочной при нашей схеме установки.

```
- : sudo cfdisk — Konsole
File Edit View Bookmarks Settings Help

Select label type
gpt
dos
sgi
sun
```

Создаём новый раздел под пространство всего **usb накопителя**.

```
- : sudo cfdisk — Konsole
File Edit View Bookmarks Settings Help

Disk: /dev/sdc
Size: 8 GiB, 8589934592 bytes, 16777216 sectors
Label: dos, identifier: 0x0ddaa7ae

>>  Device  Boot   Start      End   Sectors   Size Id Type
    Free space                2048    16777215   16775168    8G

[ New ] [ Quit ] [ Help ] [ Write ] [ Dump ]
```

Partition size: 8G

6.

May be followed by M for MiB, G for GiB, T for TiB, or S for sectors.

Тип раздела **primary**

7.

[primary] [extended]

0 primary, 0 extended, 4 free

8.

Partition type: Linux (83)

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

Did not write partition table to disk.

9.

Disk: /dev/sdc

Size: 8 GiB, 8589934592 bytes, 16777216 sectors

Label: dos, identifier: 0x0ddaa7ae

Device	Boot	Start	End	Sectors	Size	Id	Type
>> /dev/sdc1		2048	16777215	16775168	8G	83	Linux

Partition type: Linux (83)

Are you sure you want to write the partition table to disk? yes

Type "yes" or "no", or press ESC to leave this dialog.

После того как мы применили все изменения, выходим из режима редактирования и продолжаем настройку запасной загрузочной флешки.

```
Partition type: Linux (83)
[Bootable] [ Delete ] [ Resize ] [ Quit ] [ Type ] [ Help ]
[ Write ] [ Dump ]

Quit program without writing changes
```

sudo fdisk -l /dev/sdc проверяем disklabel должен быть **dos**, а device должен быть **/dev/sdc1**

```
user1@user ~: fish — Konsole
user1@user ~-> sudo fdisk -l /dev/sdc
Disk /dev/sdc: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VBOX HARDDISK
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: 0x0ddaa7ae

Device      Boot Start      End  Sectors  Size Id Type
/dev/sdc1   2048 16777215 16775168   8G 83 Linux
user1@user ~->
```

Только что созданный раздел нужно в обязательном порядке зашифровать, такие критические данные как файл ключ и заголовок недопустимо держать в незащищённом виде!!!

sudo cryptsetup --cipher=serpent-xts-plain64 --key-size=512 --hash=whirlpool luksFormat --type luks1 /dev/sdc1

шифруем раздел usb накопителя **/dev/sdc1** надёжным паролем с высокой степенью энтропии. Помните, что это первый и основной рубеж защиты ваших зашифрованных данных.

```
user1@user ~: fish — Konsole
user1@user ~-> lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda          8:0    0 50.3G  0 disk
├─cryptroot 254:2    0 50.3G  0 crypt
│ └─store-root 254:3    0  15G  0 lvm    /
│   └─store-home 254:4    0 35.3G  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
sdc          8:32    0   8G  0 disk
└─sdc1       8:33    0   8G  0 part
sr0         11:0    1 1024M  0 rom
```

```
user1@user -> sudo cryptsetup --cipher=serpent-xts-plain64 --key-size=512 --hash=whirlpool luksFormat --type luks1 /dev/sdc1
```

13.

WARNING!

=====

This will overwrite data on /dev/sdc1 irrevocably.

Are you sure? (Type 'yes' in capital letters): YES

Enter passphrase for /dev/sdc1:

Verify passphrase:

user1@user -> █

sudo cryptsetup open /dev/sdc1 cryptbootbackup открываем раздел и присваиваем ему имя **cryptbootbackup**, мы можем присвоить любое удобное и произвольное имя.

sudo mkfs.ext2 /dev/mapper/cryptbootbackup форматируем устройство cryptbootbackup в файловую систему **ext2**

sudo mount /dev/mapper/cryptbootbackup /mnt монтируем cryptbootbackup в папку **/mnt**

14.

```
user1@user -> sudo cryptsetup open /dev/sdc1 cryptbootbackup
[sudo] password for user1:
Enter passphrase for /dev/sdc1:
user1@user -> sudo mkfs.ext2 /dev/mapper/cryptbootbackup
mke2fs 1.45.6 (20-Mar-2020)
/dev/mapper/cryptbootbackup contains 'PGP Secret Key -' data
Proceed anyway? (y,N) y
Creating filesystem with 2096384 4k blocks and 524288 inodes
Filesystem UUID: c7342cca-27b5-49a7-9a2f-1985a97bc936
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

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

user1@user -> sudo mount /dev/mapper/cryptbootbackup /mnt
user1@user -> █
```

Выполняем команду **lsblk** и видим как новый раздел **cryptbootbackup** примонтирован в **/mnt**. Теперь то у нас и появилась возможность перекинуть на новую (резервную) флешку файл-ключ и заголовок от **sda**.

15.

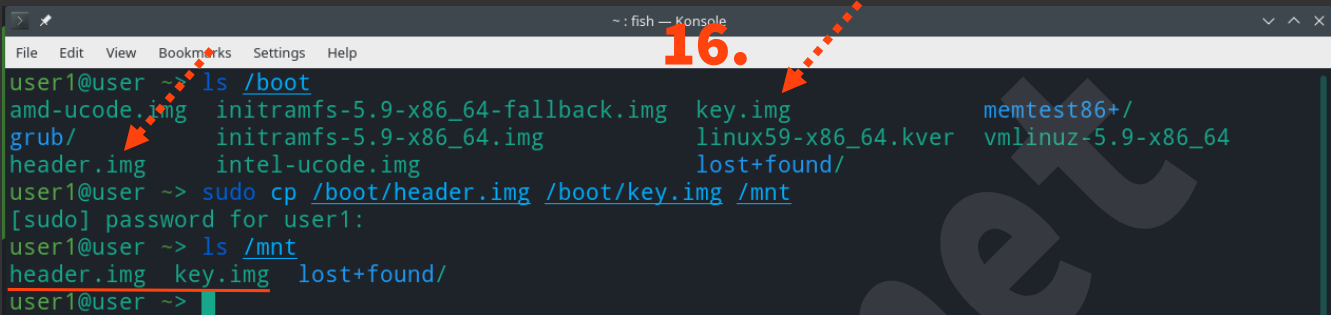
```
user1@user -> lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda                                 8:0    0 50.3G  0 disk
├─cryptroot                        254:2    0 50.3G  0 crypt
│   └─store-root                    254:3    0   15G  0 lvm    /
│       └─store-home                254:4    0 35.3G  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
sdc                                 8:32    0   8G   0 disk
├─sdc1                             8:33    0   8G   0 part
│   └─cryptbootbackup              254:1    0   8G   0 crypt /mnt
sr0                                11:0    1 1024M  0 rom
```

ls /boot как видим тут присутствуют все файлы загрузочного раздела, в том числе требуемые для переноса файлы.

sudo cp /boot/header.img /boot/key.img /mnt копируем файлы из папки **/boot** в папку **/mnt** на резервную флешку **/dev/sdc1**

ls /mnt проверяем скопировались ли файлы.

16.



```
user1@user ~$ ls /boot
amd-ucode.img  initramfs-5.9-x86_64-fallback.img  key.img  memtest86+/
grub/          initramfs-5.9-x86_64.img            linux59-x86_64.kver  vmlinuz-5.9-x86_64
header.img     intel-ucode.img                     lost+found/

user1@user ~$ sudo cp /boot/header.img /boot/key.img /mnt
[sudo] password for user1:
user1@user ~$ ls /mnt
header.img  key.img  lost+found/

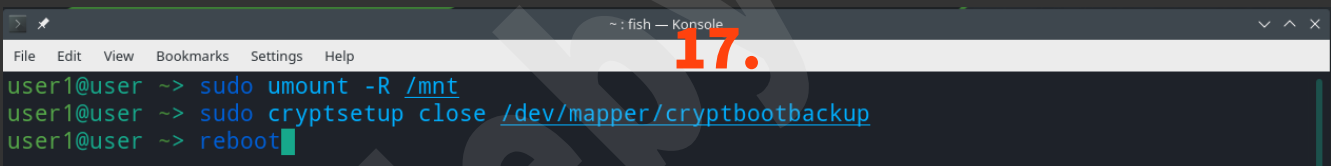
user1@user ~$
```

sudo umount -R /mnt размонтируем устройства из папки **/mnt**

sudo cryptsetup close /dev/mapper/cryptbootbackup закрываем **cryptbootbackup**

reboot перезагружаем операционную систему

17.



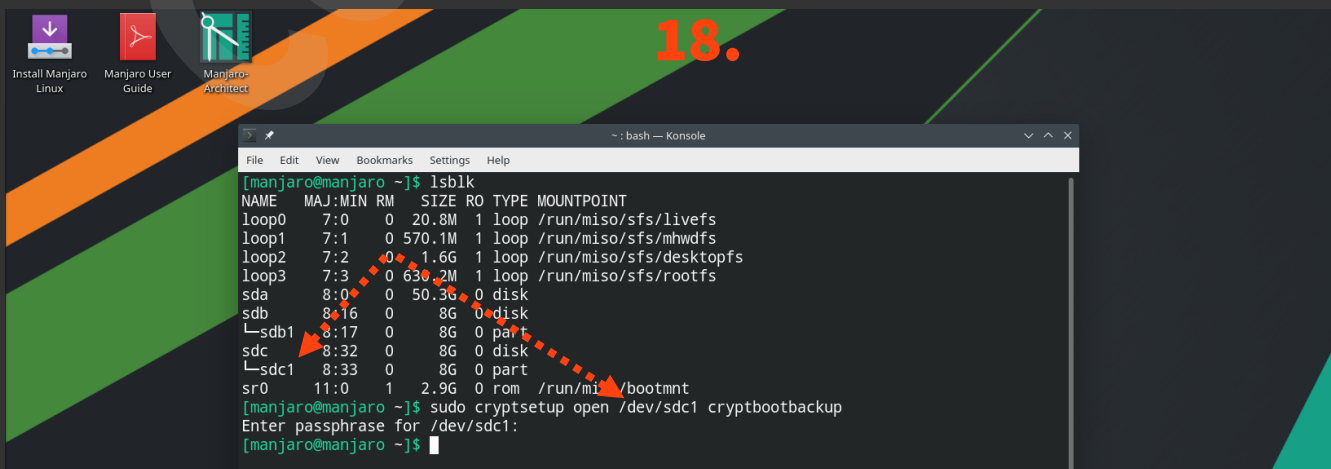
```
user1@user ~$ sudo umount -R /mnt
user1@user ~$ sudo cryptsetup close /dev/mapper/cryptbootbackup
user1@user ~$ reboot
```

Мы окончили все действия которые требовалось нам выполнить из установленной системы, а именно:

- ✓ создали зашифрованный раздел на резервной usb флешке **/dev/sdc1**
- ✓ скопировали необходимые для разблокировки **sda** файлы **header.img** и **key.img**

После данных мероприятий мы вставляем загрузочную live-флешку Manjaro KDE и перезагружаемся в неё.

18.



```
[manjaro@manjaro ~]$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
loop0 7:0 0 20.8M 1 loop /run/miso/sfs/livefs
loop1 7:1 0 570.1M 1 loop /run/miso/sfs/mhwdfs
loop2 7:2 0 1.6G 1 loop /run/miso/sfs/desktopfs
loop3 7:3 0 630.2M 1 loop /run/miso/sfs/rootfs
sda 8:0 0 50.3G 0 disk
sdb 8:16 0 8G 0 disk
└sdb1 8:17 0 8G 0 part
sdc 8:32 0 8G 0 disk
└sdc1 8:33 0 8G 0 part
sr0 11:0 1 2.9G 0 rom /run/miso/bootmnt

[manjaro@manjaro ~]$ sudo cryptsetup open /dev/sdc1 cryptbootbackup
Enter passphrase for /dev/sdc1:
[manjaro@manjaro ~]$
```

lsblk проверяем подключенные устройства к компьютеру.

sudo cryptsetup open /dev/sdc1 cryptbootbackup открываем зашифрованный раздел на шифро-флешке и присваиваем ему имя **cryptbootbackup**.

sudo mount /dev/mapper/cryptbootbackup /mnt монтируем ранее открытый раздел в папку **/mnt**

```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
19.
[manjaro@manjaro ~]$ sudo mount /dev/mapper/cryptbootbackup /mnt
[manjaro@manjaro ~]$ ls /mnt
header.img key.img lost+found
[manjaro@manjaro ~]$
```

Теперь на смонтированном устройстве у нас стали доступны ключевые файлы и мы можем с помощью их открыть диск **sda**

sudo cryptsetup open /mnt/key.img lukskey открываем ключ и присваиваем имя **lukskey**

```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
20.
[manjaro@manjaro ~]$ sudo mount /dev/mapper/cryptbootbackup /mnt
[manjaro@manjaro ~]$ ls /mnt
header.img key.img lost+found
[manjaro@manjaro ~]$ sudo cryptsetup open /mnt/key.img lukskey
Enter passphrase for /mnt/key.img:
No key available with this passphrase.
Enter passphrase for /mnt/key.img:
No key available with this passphrase.
Enter passphrase for /mnt/key.img:
[manjaro@manjaro ~]$ lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0                 7:0    0  20.8M  1 loop /run/miso/sfs/livefs
loop1                 7:1    0  570.1M  1 loop /run/miso/sfs/mhwdfs
loop2                 7:2    0    1.6G  1 loop /run/miso/sfs/desktopfs
loop3                 7:3    0  630.2M  1 loop /run/miso/sfs/rootfs
loop4                 7:4    0    32M   0 loop
└─lukskey             254:1    0    16M   0 crypt
sda                   8:0     0  50.3G  0 disk
sdb                   8:16    0     8G   0 disk
└─sdb1                8:17    0     8G   0 part
sdc                   8:32    0     8G   0 disk
└─sdc1                8:33    0     8G   0 part
   └─cryptbootbackup  254:0    0     8G   0 crypt /mnt
sr0                   11:0    1    2.9G  0 rom  /run/miso/bootmnt
[manjaro@manjaro ~]$
```

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

открываем зашифрованный жёсткий диск **/dev/sda**. Мы видим как после открытия **cryptroot** у нас также открылись разделы lvm **store-root** и **store-home**.


```
21.
[manjaro@manjaro ~]$ sudo cryptsetup open --header=/mnt/header.img --key-file=/dev/mapper/lukskey --keyfile-offset=9437 --keyfile-size=8192 /dev/sda cryptroot
[manjaro@manjaro ~]$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0                               7:0      0  20.8M 1 loop /run/miso/sfs/livefs
loop1                               7:1      0 570.1M 1 loop /run/miso/sfs/mhwdfs
loop2                               7:2      0   1.6G 1 loop /run/miso/sfs/desktopfs
loop3                               7:3      0 630.2M 1 loop /run/miso/sfs/rootfs
loop4                               7:4      0    32M 0 loop
└─ lukskey                          254:1      0   16M 0 crypt
sda                                 8:0      0  50.3G 0 disk
├─ cryptroot                       254:2      0  50.3G 0 crypt
│   └─ store-root                  254:3      0   15G 0 lvm
│       └─ store-home              254:4      0  35.3G 0 lvm
sdb                                 8:16      0    8G 0 disk
└─ sdb1                            8:17      0    8G 0 part
sdc                                 8:32      0    8G 0 disk
└─ sdc1                            8:33      0    8G 0 part
    └─ cryptbootbackup             254:0      0    8G 0 crypt /mnt
sr0                                 11:0      1   2.9G 0 rom  /run/miso/bootmnt
[manjaro@manjaro ~]$
```

sudo cryptsetup close lukskey закрываем **lukskey**

sudo umount /mnt размонтируем подключенные устройства из папки **/mnt**

```
22.
[manjaro@manjaro ~]$ sudo cryptsetup close lukskey
[manjaro@manjaro ~]$ sudo umount /mnt
[manjaro@manjaro ~]$
```

sudo mount /dev/store/root /mnt монтируем корневой каталог в папку **/mnt**

sudo mount /dev/store/home /mnt/home монтируем домашний каталог в папку **/mnt/home**

sudo mount /dev/mapper/cryptboot /mnt/boot монтируем нашу шифро-флешку **cryptbootbackup** в загрузочную папку **/mnt/boot**

```
23.
[manjaro@manjaro ~]$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0                               7:0      0  20.8M 1 loop /run/miso/sfs/livefs
loop1                               7:1      0 570.1M 1 loop /run/miso/sfs/mhwdfs
loop2                               7:2      0   1.6G 1 loop /run/miso/sfs/desktopfs
loop3                               7:3      0 630.2M 1 loop /run/miso/sfs/rootfs
sda                                 8:0      0  50.3G 0 disk
├─ cryptroot                       254:2      0  50.3G 0 crypt
│   └─ store-root                  254:3      0   15G 0 lvm
│       └─ store-home              254:4      0  35.3G 0 lvm
sdb                                 8:16      0    8G 0 disk
└─ sdb1                            8:17      0    8G 0 part
sdc                                 8:32      0    8G 0 disk
└─ sdc1                            8:33      0    8G 0 part
    └─ cryptbootbackup             254:0      0    8G 0 crypt
sr0                                 11:0      1   2.9G 0 rom  /run/miso/bootmnt
[manjaro@manjaro ~]$ sudo mount /dev/store/root /mnt
[manjaro@manjaro ~]$ sudo mount /dev/store/home /mnt/home/
[manjaro@manjaro ~]$ sudo mount /dev/mapper/cryptbootbackup /mnt/boot/
[manjaro@manjaro ~]$
```

```
24.
[manjaro@manjaro ~]$ ls /mnt
bin boot dev etc home hostlvm lib lib64 lost+found mnt opt proc root run sbin snap srv sys tmp usr var
[manjaro@manjaro ~]$ ls /mnt/boot/
header.img key.img lost+found
[manjaro@manjaro ~]$
```

Не забываем проверить правильность монтирования разделов!!!

```
25.
[manjaro@manjaro ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0       7:0      0   20.8M 1 loop /run/miso/sfs/livefs
loop1       7:1      0  570.1M 1 loop /run/miso/sfs/mhwdfs
loop2       7:2      0    1.6G 1 loop /run/miso/sfs/desktopfs
loop3       7:3      0  630.2M 1 loop /run/miso/sfs/rootfs
sda         8:0      0   50.3G 0 disk 
├─cryptroot 254:2    0   50.3G 0 crypt 
├─store-root 254:3    0   15.6G 0 lv /mnt
└─store-home 254:4    0   35.3G 0 lv /mnt/home
sdb         8:16     0     8G 0 disk 
├─sdb1       8:17     0     8G 0 part 
├─sdc        8:32     0     8G 0 disk 
└─sdc1       8:33     0     8G 0 part 
    └─cryptbootbackup 254:0    0     8G 0 cry /mnt/boot
sr0         11:0     1    2.9G 0 rom  /run/miso/bootmnt
[manjaro@manjaro ~]$
```

manjaro-chroot /mnt входим в окружение **chroot** для завершения настроек

```
26.
[manjaro@manjaro ~]$ manjaro-chroot /mnt
sh-5.1#
```

sudo pacman -Sy обновляем базу пакетов

sudo pacman -S linux59 устанавливаем ядро на **/dev/sdc1**. В моём случае я ставлю 59е ядро, но к моменту прохождения курса всё может поменяться. По этой причине версию ядра выбираете самостоятельно.

```
27.
sh-5.1# sudo pacman -Sy
:: Synchronizing package databases...
core is up to date
extra is up to date
community is up to date
multilib is up to date
sh-5.1#
```

```
28.
sh-5.1# sudo pacman -S linux59
warning: linux59-5.9.16-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) linux59-5.9.16-1
Total Installed Size: 154.96 MiB
Net Upgrade Size: 0.00 MiB

:: Proceed with installation? [Y/n] Y
(1/1) checking keys in keyring [#####] 100%
(1/1) checking package integrity [#####] 100%
(1/1) loading package files [#####] 100%

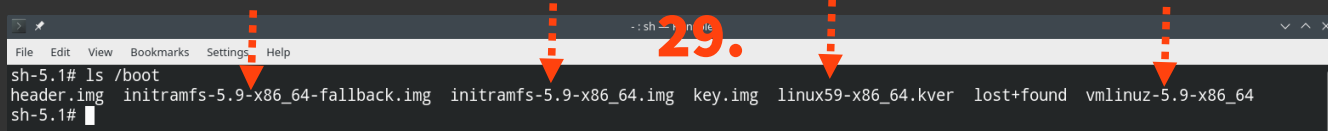
==> Image generation successful
(4/5) Updating Grub-Bootmenu
/usr/bin/grub-editenv: error: cannot open `../boot/grub/grubenv.new': No such file or directory.
/usr/bin/grub-mkconfig: line 259: /boot/grub/grub.cfg.new: No such file or directory
error: command failed to execute correctly
(5/5) Restore Linux kernel modules
ls: cannot access 'backup': No such file or directory

==> Warning:
-> Kernel has been updated. Modules of the current kernel
-> have been backed up so you can continue to use your
-> computer. However, the new kernel will only work
-> at next boot.

sh-5.1#
```

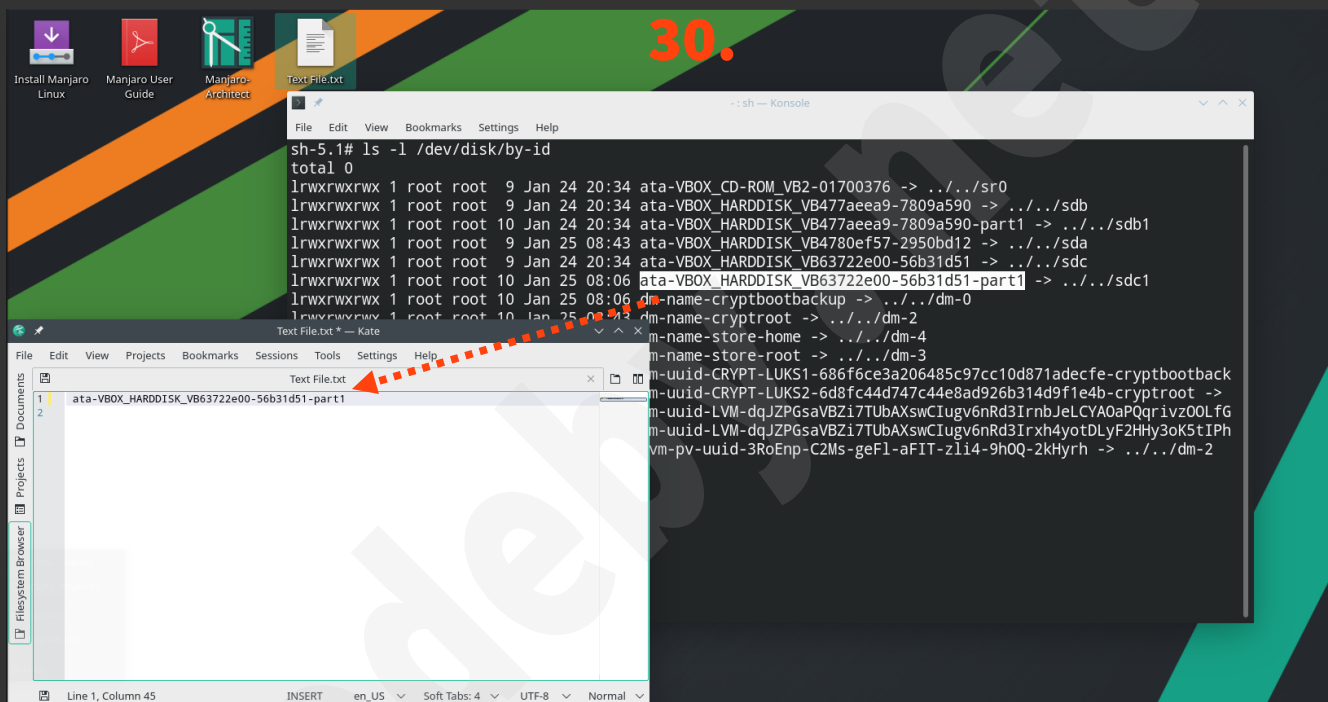
На изображении сверху я выделил возникающую ошибку в самом окончании установки ядра, не обращаем на это внимание. Просто ядро жалуется что отсутствует папка grub необходимая для загрузки операционной системы. Этот вопрос мы решим чуть позже

ls /boot проверяем наличие новых файлов после установки ядра

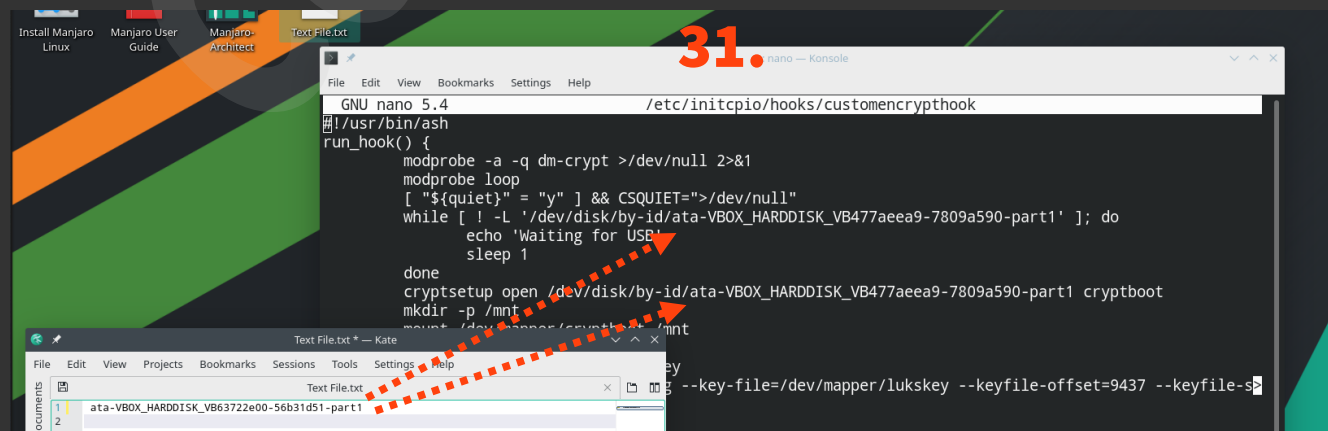


```
sh-5.1# ls /boot
header.img  initramfs-5.9-x86_64-fallback.img  initramfs-5.9-x86_64.img  key.img  linux59-x86_64.kver  lost+found  vmlinuz-5.9-x86_64
sh-5.1#
```

ls -l /dev/disk/by-id для удобства копируем идентификатор новой загрузочной флешки в текстовый документ на рабочем столе. В моем случае это **sdc1**



nano /etc/initcpio/hooks/customencrypthook заменяем идентификатор старой флешки на идентификатор новой. Таким образом давая системе понять какая из них загрузочная.



Не забываем в обязательном порядке в файле **customencrypthook** указать верное название раздела **/dev/sdc1** в нашем случае это **cryptbootbackup**

```
GNU nano 5.4 /etc/initcpio/hooks/customencrypthook Modified
#!/usr/bin/ash
run_hook() {
    modprobe -a -q dm-crypt >/dev/null 2>&1
    modprobe loop
    [ "${quiet}" = "y" ] && CSQUIET=">/dev/null"
    while [ ! -L '/dev/disk/by-id/ata-VBOX_HARDDISK_VB63722e00-56b31d51-part1' ]; do
        echo 'Waiting for USB'
        sleep 1
    done
    cryptsetup open /dev/disk/by-id/ata-VBOX_HARDDISK_VB63722e00-56b31d51-part1 cryptbootbackup
    mkdir -p /mnt
    mount /dev/mapper/cryptbootbackup /mnt
    cd /mnt
    cryptsetup open key.img lukskey
    cryptsetup --header header.img --key-file=/dev/mapper/lukskey --keyfile-offset=9437 --keyfile-size=8192 o
    cd /
    cryptsetup close lukskey
    umount /mnt
}
```

nano /etc/fstab открываем файл и смотрим на последнюю строку. Нам нужно заменить UUID раздела **/boot**, поскольку данный UUID на старой и новой флешке будут отличаться.

Повторюсь: если этого не сделать то система попросту не загрузится.

Чтобы узнать данный идентификатор нужно на рабочем столе открыть новый терминал и выполнить команду:

lsblk -f после чего находим в списке раздел **cryptbootbackup** с файловой системой ext2. Именно этот идентификатор нам нужно вставить в файл **fstab**. Замечу ибо это важно. Не идентификатор раздела sdc1, а именно уже открытый контейнер **cryptbootbackup**!!!

```
GNU nano 5.4 /etc/fstab Modified
# /dev/mapper/store-root
UUID=5ec54677-ecc4-444d-bf5d-21d620b1120d / ext4
# /dev/mapper/store-home
UUID=91a3301a-9a14-440e-b100-63875f285a0d /home ext4
# /dev/mapper/cryptboot
UUID=c7342cca-27b5-49a7-9a2f-1985a97bc936 /boot ext2
```

```
[manjaro@manjaro ~]$ lsblk -f
NAME        FSTYPE     FSVER    LABEL          UUID                                  FSAVAIL FSUSE% MOUNTPOINT
loop0       squashfs   4.0                               3RoEnp-C2Ms-geF1-aFIT-zli4-9hOQ-2kHyrh
loop1       squashfs   4.0                               5ec54677-ecc4-444d-bf5d-21d620b1120d
loop2       squashfs   4.0                               91a3301a-9a14-440e-b100-63875f285a0d
loop3       squashfs   4.0                               5d2445cc-7477-4b73-b045-4b2a42054aeb
sda
├─cryptroot  LVM2_member LVM2 001 3RoEnp-C2Ms-geF1-aFIT-zli4-9hOQ-2kHyrh
│ └─store-root ext4      1.0  5ec54677-ecc4-444d-bf5d-21d620b1120d  3.9G  68% /mnt
│ └─store-home ext4      1.0  91a3301a-9a14-440e-b100-63875f285a0d  32.6G  0% /mnt/home
├─sdb
│ └─sdb1     crypto_LUKS 1  5d2445cc-7477-4b73-b045-4b2a42054aeb
├─sdc
│ └─sdc1     crypto_LUKS 1  686f6ce3-a206-485c-97cc-10d871adecfe
│   └─cryptbootbackup ext2      1.0  c7342cca-27b5-49a7-9a2f-1985a97bc936  7.3G  2% /mnt/boot
sr0         iso9660     Joliet Extensio MANJARO_KDE_2012 2020-10-19-13-06-32-00 0 100% /run/miso/bootmnt
```

Не забудьте поменять **/dev/mapper/cryptboot** на **/dev/mapper/cryptbootbackup**. На изображении внизу приведён готовый файл **fstab**.

```
34.
GNU nano 5.4 /etc/fstab
# /dev/mapper/store-root
UUID=5ec54677-ecc4-444d-bf5d-21d620b1120d / ext4 rw,relatime 0 0

# /dev/mapper/store-home
UUID=91a3301a-9a14-440e-b100-63875f285a0d /home ext4 rw,relatime 0 0

# /dev/mapper/cryptbootbackup
UUID=c7342cca-27b5-49a7-9a2f-1985a97bc936 /boot ext2 rw,relatime 0 0
```

После сохранения всех документов нам потребуется обновить initramfs.

mkinitcpio -p /etc/mkinitcpio.d/linux59.preset где вместо 59 указываем то ядро, которое вы установили. Обратите внимание чтобы при генерировании не возникло никаких ошибок, в особенности с модулем **customcrypthook**

```
35.
sh-5.1# mkinitcpio -p /etc/mkinitcpio.d/linux59.preset
==> Building image from preset: /etc/mkinitcpio.d/linux59.preset: 'default'
-> -k /boot/vmlinuz-5.9-x86_64 -c /etc/mkinitcpio.conf -g /boot/initramfs-5.9-x86_64.img
==> Starting build: 5.9.16-1-MANJARO
-> Running build hook: [base]
-> Running build hook: [udev]
-> Running build hook: [autodetect]
-> Running build hook: [keymap]
-> Running build hook: [modconf]
-> Running build hook: [block]
-> Running build hook: [customcrypthook]
-> Running build hook: [lvm2]
-> Running build hook: [filesystems]
-> Running build hook: [keyboard]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-5.9-x86_64.img
==> Image generation successful
==> Building image from preset: /etc/mkinitcpio.d/linux59.preset: 'fallback'
-> -k /boot/vmlinuz-5.9-x86_64 -c /etc/mkinitcpio.conf -g /boot/initramfs-5.9-x86_64-fallback.img -S aut
```

Самым последним шагом является установка загрузчика **GRUB**. Загрузчик grub нужно поставить в главную загрузочную запись новой флешки. Так если раздел **cryptbootbackup** у нас **sdс1**, то загрузчик нужно ставить в **sdс**. Это будет единственная незашифрованная часть нашей операционной системы, однако это не суть важно.

grub-install --recheck /dev/sdc устанавливаем загрузчик

grub-mkconfig -o /boot/grub/grub.cfg обновляем конфигурационный файл.

```
36.
sh-5.1# grub-install --recheck /dev/sdc
Installing for i386-pc platform.
Installation finished. No error reported.
```

На возникающие ошибки связанные с lvmata не обращаем внимания!

```
sh-5.1# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
Found linux image: /boot/vmlinuz-5.9-x86_64
Found initrd image: /boot/initramfs-5.9-x86_64.img
Found initrd fallback image: /boot/initramfs-5.9-x86_64-fallback.img
WARNING: Failed to connect to lvmata. Falling back to device scanning.
WARNING: Failed to connect to lvmata. Falling back to device scanning.
done
sh-5.1#
```

exit выходим из окружения chroot

umount -R /mnt размонтируем устройства с которыми работали из папки /mnt

reboot перезагружаем компьютер

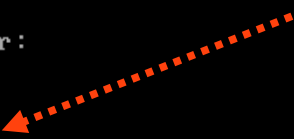
```
sh-5.1# exit
exit
[manjaro@manjaro ~]$ umount -R /mnt
umount: /mnt/home: must be superuser to unmount.
[manjaro@manjaro ~]$ sudo umount -R /mnt
[manjaro@manjaro ~]$
```

VirtualBox temporary boot device selection

Detected Hard disks:

AHCI controller:

- 1) Hard disk
- 2) Hard disk
- 3) Hard disk



После перезагрузки мы должны войти в **BOOT Menu** компьютера и выбрать загрузку с нового usb-накопителя. В моём случае это Hard Disk под номером 3, поскольку для примеров я использую виртуальную машину virtualbox.

Вводим 3 раза пароль для загрузки операционной системы

```
Attempting to decrypt master key... 40.  
Enter passphrase for hd2,msdos1 (686f6ce3a206485c97cc10d871adecfe):
```

После загрузки рабочего стола в терминале вводим команду **lsblk** и смотрим название раздела, который имеет точку монтирования **/boot**. На изображении внизу это **cryptbootbackup**, который мы с вами и настраивали. Это значит, что резервную флешку мы создали верно. Стоит оговориться что одновременно у вас будет работать только одна флешка, вторая же при загрузке будет давать сбой. Связано это в первую очередь с тем, что накопители имеют разные названия, а главные идентификаторы **id** и **uuid**. Следовательно в файлах **customencrypthook** и **fstab** флешки будут иметь совершенно разные значения. Но наша задача заключалась не в том, чтобы сделать две одновременно работающие флешки. Наша задача сделать резервную флешку, да резервная флешка не будет самостоятельно загружаться при поломке основной, но она позволит нам открыть диск **sda**, войти в окружение **chroot** и подправить два файла которые нуждаются в правильных идентификаторах. И после этого резервная флешка станет основной.

Ещё раз повторим: "Резервная зашифрованная флешка не будет работать одновременно с основной загрузочной. Но резервный накопитель позволит восстановить полную работоспособность системы в том случае, когда основной по различным причинам выйдет из строя."

```
user1@user -> lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	50.3G	0	disk	
└─cryptroot	254:2	0	50.3G	0	crypt	
└─┬─store-root	254:3	0	15G	0	lvm	/
└─└─store-home	254:4	0	35.3G	0	lvm	/home
sdb	8:16	0	8G	0	disk	
└─sdb1	8:17	0	8G	0	part	
sdс	8:32	0	8G	0	disk	
└─sdс1	8:33	0	8G	0	part	
└─cryptbootbackup	254:0	0	8G	0	crypt	/boot
sr0	11:0	1	2.9G	0	rom	