Диск Диску рознь. Вроде диск, а вроде нет.

Во 2-й статье в конце мы говорили, что есть ещё один способ создание образа из папки, который гарантирует целостность ваших данных.

Сделать это можно с помощью утилиты dd. Фактически, это аналог утилиты копирования файлов cp только для блочных данных.

Утилита просто переносит по одному блоку данных указанного размера с одного места в другое.

Поскольку в Linux все, в том числе, устройства, считается файлами, вы можете переносить устройства в файлы и наоборот.

С помощью различных опций утилиты можно повлиять на размер блока, а это, в свою очередь, уже влияет на скорость работы программы.

Однако, она работает только с блочными устройствами, т.е. преобразовать папку напрямую в образ нельзя.

Тогда как dd может помочь?

Всё достаточно просто. Создадим виртуальный RAW диск и на весь диск одну файловую систему. Отформатируем, примонтируем, скопируем туда наши данные. И уже как с блочного устройство loop0 просто создадим образ диска.

Единственное, что нам необходимо будет помнить, что в итоге это всё-таки не совсем именно ISO-образ. Да, это образ, но он как-бы содержит в себе таблицу разделов нашего виртуального диска.

Мы можем убедиться в этом, если примонтируем полученный ISO-файл и посмотрим на тип раздела loop устройства. Не все архиваторы могут прочитать такой образ. Если при использовании genisoimage по сути по большей части получается архив с форматом данных, то здесь именно образ виртуального диска.

При чтении с помощью 7z мы увидим там 2 новых папки - [SYS] и lost+found.

Однако, его всё равно можно открыть и в Linux и в Windows. 7zip прекрасно читает. Консольный 7z Linux тоже.

#

$ dd if=/dev/zero of=raw.img bs=1M count=2560

$ mkfs -t ext4 raw.img

$ mkdir -p ./raw && sudo mount -t auto -o loop raw.img ./raw

$ sudo chown mikl:users ./raw && sudo chmod 777 ./raw

$ sudo umount -l ./raw && sudo rm -rf ./raw

#

Чтобы конвертировать ISO образ даже такого диска как выше в архив существует как минимум 2 способа:

1) Необходимо примонтировать диск, заархвировать полученные данные и размонтировать диск.

2) Образ можно просто распаковать с помощью 7z во временную директорию, удалить каталоги [SYS] и lost+found, упаковать в архив, и удалить временную директорию.

К сожалению других вариантов нет.

#

Во 2-м способе можно немного поиграться с архиватором 7z. В каждом из вариантов есть свои преимущества и недостатки.

Для примера пусть будет образ image.iso, временные папки script (или temp), архив archive.7z:

1) $ mkdir -p script && 7z x image.iso -o./script/ && rm -rf ./script/[SYS] ./script/lost+found && tar -czf archive.tar.gz script/ && rm -rf ./script

2) $ mkdir -p temp && mv image.iso ./temp/ && cd ./temp && 7z x image.iso && 7z a '-x!image.iso' '-x![SYS]' '-x!lost+found' -t7z -mx7 ../archive.7z && mv image.iso ../ && cd ../ && rm -rf ./temp/

3) $ 7z a '-x!disk/[SYS]' '-x!disk/lost+found' -t7z -mx7 archive.7z ./disk

4) $ cd ./disk && 7z a '-x![SYS]' '-x!lost+found' -t7z -mx7 ../archive.7z && cd ../

Обратите внимание что в любом из вариантов исключение или удаление sys и lost+found происходит по относительному пути. Т.е. при их наличии в одной из поддиректорий - их тоже надо указывать.

#

Переходим к созданию более правильного RAW диска, при конвертации которого, Windows сможет его корректно прочитать.

Но оставим и предыдущий вариант с одной файловой системой на весь диск - для тестов. Он прекрасно подходит для QEMU и + мы сможем работать с таким диском, без каких-либо дополнительных утилит.

Если нам нужно использовать последний правильный RAW-диск не только в Linux, но и в Windows, тогда мы можем либо конвертировать, либо аналогично создать сразу подходящий диск без конвертации.

Но пока создадим именно первый вариант диска. Для создания таблицы разделов и самих разделов будем использовать parted.

В ней необходимо помнить 2 важных правила:

1) При создании нового раздела смещать его относительно предыдущего на 1MB или 1MiB.

2) Указывать размеры можно не только в MB(kB,GB, MiB...), но и в процентах.

$ dd if=/dev/zero of=raw.img bs=4096k count=640

$ parted -s raw.img print

$ parted -s raw.img mklabel msdos

или parted -s raw.img mklabel gpt

$ parted -s raw.img mkpart primary ntfs 1MiB 100%

#

Если у вас GPT таблица, можете, при необходимости, создать ESP раздел для UEFI вот так:

parted -s DEVICE mkpart ESP fat32 1MiB 513MiB

А для указания загрузочного раздела в любой из таблиц, воспользуйтесь такой командой:

parted -s DEVICE set 1 boot on

Где DEVICE устройство с которым вы работаете. Это может быть, как у меня образ диска, а может и указанный диск. Именно диск, а не раздел диска!

Например, /dev/sdb или /dev/sdc. Далее по скриншотам будет более понятно на конкретных примерах.

#

$ parted -s raw.img print

У всех виртуальных дисков есть один недостаток. При создании таблицы разделов изменений сразу может быть не видно, пока диск не будет пере-подключен!

#

С таким RAW-диском уже не получится работать как с одно-файловым и монтировать в виде лазерного диска в loop0.

Для этого нам понадобится утилита для блочных устройств, а для файловых систем NTFS и FAT32 соответственно 2 других утилиты.

$ sudo pacman -S nbd ntfs-3g dosfstools --noconfirm

$ sudo modprobe nbd

#

Подключаем, форматируем, отключаем.

$ sudo qemu-nbd -c /dev/nbd0 ./raw.img

Вообще мы можем таким образом подключить любой диск - VDI, VHD, VMDK, VHDX, RAW.

И затем уже создавать на них таблицу разделов с помощью cfdisk, cgdisk, gparted, parted…

Для Linux и qemy-nbd никакой разницы нет.

$ lsblk -o NAME,MODEL,TYPE,FSTYPE,SIZE,MOUNTPOINT /dev/nbd0

$ mkfs -t ntfs /dev/nbd0p1

Обратите внимание, что для файловой системы NTFS устройство обязательно должно быть блочным. Иначе mkfs выдаст ошибку.

Нельзя просто взять и отформатировать весь диск только под NTFS. Нужен хотя бы один раздел, который и будет отформатирован в последний.

Из вышеуказанной особенности виртуальных дисков, пере-подключаем, копируем туда данные. Отключаем и конвертируем в другой формат.

Для отключения воспользуйтесь следующей командой:

$ sudo qemu-nbd -d /dev/nbd0

#

Проверяем, конвертируем.

$ parted -s raw.img print

$ qemu-img convert -p -f raw -O vdi raw.img raw_vdi.vdi

raw(img) vpc(vhd) vdi(vdi) vhdx(vhdx) qcow2(qcow)

Для разных форматов просто берем указанные перед скобками значения, а расширения подставляем из скобок.

$ qemu-img convert -p -f raw -O vpc raw.img raw_vhd.vhd

Подключаем через qemu-nbd, проверяем. Не забываем по окончании работы с диском сначала размонтировать их, затем отключить в qemu-nbd.

Иначе, у вас начнутся большие неприятности. Просто посыпятся куча ошибок, которые уже не удастся исправить без перезагрузки. Запомните порядок. **В каком подключали, в таком в обратном порядке и отключаете.**

#

Мы могли бы сразу создать образ виртуального диска в любом из форматов, не особо заморачиваясь.

$ qemu-img create -f vpc -o size=4G ./image_vhd.vhd

или так: qemu-img create -f qcow2 hard.qcow 30G

raw(img) vpc(vhd) vdi(vdi) vhdx(vhdx) qcow2(qcow)

Аналогично подставляем необходимые параметры.

Далее просто подключайте образ к виртуальной машине и всё. Уже внутри машины силами самих ОС будут созданы необходимые разделы, при необходимости.

#

Единственный недостаток такого метода. Нельзя работать только с одним единственным диском - qcow от QEMU.

#

Вот пример создания простого VHD-диска. При таком методе, права на директорию, куда монтируется диск, задавать не обязательно.

$ qemu-img create -f vpc -o size=4G ./image_vhd.vhd

```
$ sudo qemu-nbd -c /dev/nbd0 ./image_vhd.vhd

$ parted -s /dev/nbd0 mklabel msdos

$ parted -s /dev/nbd0 mkpart primary ntfs 1MB 2561MB

$ parted -s /dev/nbd0 mpart primary ext4 2562MB 100%

$ sudo qemu-nbd -d /dev/nbd0

$ sudo qemu-nbd -c /dev/nbd0 ./image_vhd.vhd

$ mkfs -t ntfs /dev/nbd0p1

$ mkfs -t ext4 /dev/nbd0p2

$ sudo qemu-nbd -d /dev/nbd0

$ sudo qemu-nbd -c /dev/nbd0 ./image_vhd.vhd

$ lsblk -o NAME,MODEL,TYPE,FSTYPE,SIZE,MOUNTPOINT /dev/nbd0

$ parted -s /dev/nbd0 print

$ id -g `$whoami`

$ id -u `$whoami`

$ sudo mount -t auto -o,rw,gid=985,uid=1000 /dev/nbd0p1 ./disk

#
```

А по окончании работы с диском:

```
$ sudo umount -l ./disk

$ sudo qemu-nbd -d /dev/nbd0

#
```

Вот теперь все диски можно подключить в виртуальных машинах и проверить на работоспособность, а также сделать тоже самое на Windows.

Для VirtualBox ничего делать не требуется.

Для Qemu можно поиграться с настройками. Так у нас будет больше преимуществ.

Посмотрим на все устройства и процессоры.

$ qemu-system-x86_64 -device help > device.txt

$ qemu-system-x86_64 -cpu help > cpu.txt

Подключить диски будем вот по такому принципу:

-drive format=vpc,file=image.vhd

vpc(file.vhd) vdi(file.vdi) vhdx(file.vhdx)

#

Создадим короткий скрипт, чтобы не копи-пастить команды в консоль.

```
#!/bin/bash

qemu-system-x86_64 \

-enable-kvm \

-cpu host \

-smp cores=1 \

-m 1024 \

-machine q35 \

-device intel-iommu \

-vga virtio \

-device e1000,netdev=wan \

-netdev user,id=wan \
```

-boot menu=on \

-cdrom linuxmint-20.2-mate-64bit.iso \

-drive format=raw,file=raw.img \

-drive format=vpc,file=image_vhd.vhd \

-drive format=vdi,file=image_vdi.vdi

-hda hard.qcow

#-L . \

#-drive file=/usr/share/ovmf/x64/OVMF_CODE.fd,if=pflash,format=raw,unit=0,readonly=on

или

#-drive file=OVMF_CODE.fd,if=pflash,format=raw,unit=0,readonly=on

Последние 5 строчек - это бонус для возможности запуска QEMU в режиме UEFI. Но, только если у вас установлен edk2-ovmf.

Однако, вы можете скачать готовый пакет и вытащить из него буквально 2 файла - один для 32 бит, другой для 64 - OVMF_CODE.fd.

Ну и конечно же не забываем про подключение к Qemu интернета.

#

В самом начале заголовка статьи указана VeraCrypt не спроста.

По сути это ведь такой же контейнер, который представляет из себя образ диска, только шифрованный.

В Windows VeraCrypt может создавать шифрованные диски со следующими файловыми системами**:** *NTFS, FAT, exFAT*.

В Linux: *FAT, NTFS, Btrfs, EXT2, EXT3, EXT4*.

Таким образом созданный диск в **Linux**, не всегда можно подключить в **Windows**. Хотя наоборот - возможно.

#

Всегда думайте какой диск вам лучше использовать для ваших данных.

Для любого диска всегда помните об ограничениях файловых систем и вы никогда не потеряете свои данные.

#

Надеюсь я вас хоть немного заинтересовал.

Всем до встречи. Пока-пока!