Лекция 7. Семинар 7. Предоставление клиенту блочного устройства от сервера

Блочные устройства

В Linux любая файловая система строится на блочном устройстве.

Блочное устройство - специальное устройство, которое обладает свойствами:

1. состоит из блоков с одинаковым размерами (у разных устройств могут быть разные размеры);
2. можно получить произвольный доступ к любому блоку (соответсвенно, есть команды `write`, `read`, `seek`).

Пример блочного устройства - жёсткий диск.
`ls -l /dev/sda`
`brw--rw----` - блочное устройство помечается флажком "b", первый флаг отображает тип файла.

Все блочные устройства в Linux отображаются как специальный блочный файл (block special file), это можно посмотреть, например утилитой `stat`.
`stat /dev/sda`

`lsblk` - показывает блочные устройства, которые есть в системе.

Блочные устройства бывают разных типов, например: loop, disk, part, rom. Блочное устройство может реализовываться несколькими типами драйверов (то есть модулями, которые есть в ядре Linux), и из-за этого блочные устройства могут иметь разный тип. Каждому типу соответствует мажорное число (`MAJ`), минорное число (`MIN`) служит в качестве порядкого номера устройства определённого типа.

`dd` - утилита, которая выполняет поблочное копирование данных из одного места в другое на двоичном уровне.
`dd if=источник_копирования of=место_назначения параметры`

Некоторые параметры:

- `bs` - указывает сколько байт читать и записывать за один раз (по умолчанию 512 байт);

- `count` - скопировать указанное количество блоков, размер одного блока указывается в параметре `bs`;
- `seek` - пропустить указанное количество байт в начале устройства для чтения.

Сделаем из файла блочное устройство

`dd if=/dev/zero of=test.img bs=1024 count=1024` - получили регулярный (обычный) файл test.img размером 1 МБ, заполненный нулями.

`losetup` - утилита, позволяющая контролировать блочные устройства.

`sudo losetup -a` - показывает все имеющиеся блочные устройства типа loop (их статус).

`sudo losetup -f test.img` - находит первое неиспользующееся блочное устройство типа loop и привязывает файл test.img к нему (например к dev/loop7).

`sudo mkfs.vfat /dev/loop7` - отформатировать блочное устройство. Теперь test.img уже не является пустым файлом, а распознаётся как устройство.

`sudo mount /dev/loop7 /mnt` - смонтировать виртуальное блочное устройство, то есть подключить к какой-то ветке (смонтировали в /mnt).

`nano /mnt/read.me` - добавили в /mnt какой-то текстовый файл.

`sudo umount /mnt` - размонтировать.

`sudo losetup -d /dev/loop7` - отсоединить файл test.img от блочного устройства loop7.

`strings test.img` - выводит строки, содержащиеся в двочном файле. Видим, что файл содержит строки, которые были записаны в /mnt/read.me.

Экспорт блочного устройства из сервера клиенту

iSCSI (Internet Small Computer System Interface) — протокол, который базируется на TCP/IP и разработан для установления взаимодействия и управления системами хранения данных, серверами и клиентами.

- **initiator** (по сути клиент) — тот, кто устанавливает соединение с target (с экспортируемым объектом). Чаще всего это узел (в общем случае), осуществляет ввод/вывод на блочные устройства.
- **Портал** — группа таргетов, которые анонсируются вместе. Чаще всего один узел хранения — один портал.

- **IQN** — полное имя участника взаимодействия. На практике существует iqn у инициатора и у таргета.
- **LUN (Logical Unit Number)** — номер объекта внутри таргета. Target состоит из кусочков, называемых LUN. Ближайшим аналогом является раздел диска или отдельный том.

Серверная часть

`tgt` - **The Linux target framework**. Пакет, который используется в качестве сервера. Это специальный демон, специальное ПО, которое с одной стороны будет цепляться к блочному устройству на этом компьютере и, когда инициатор подключится, будет передавать информацию в блоках этому инициатору.

`sudo apt install tgt` - установка tgt.
Рекомендация: не забывать перед установкой выполнять `sudo apt update`.

`dpkg -L tgt | less` - показывает те файлы, которые установил пакет tgt.

Таргет

Сначала нужно получить то блочное устройство, которое будем расшаривать. Можно подключить виртуальный жёсткий диск, можно сделать образ файла (выберем этот вариант).
`sudo mkdir storage` - создаём каталог storage
`sudo dd if=/dev/zero of=/storage/lun0.img bs=1M count=2048` - создаём в storage файл lun0.img размером 2 ГБ.
`sudo ls -lh /storage/lun0.img` - проверили, что такой файл с таким размером создан.

`cd /etc/tgt/` - перейдём сюда, чтобы изменить конфигурацию targets.conf.
`less targets.conf` - видим информацию о том, что targets.conf включает в себя все конфигурационные файлы, которые содержатся в /etc/tgt/conf.d/. Значит, чтобы создать собственную конфигурацию, надо в этом каталаге создать свой конфигурационный файл и его отредактировать нужным образом.
`cd conf.d/` - переходим в /etc/tgt/conf.d/.
`sudo nano test.conf`

```
<target iqn.2021-10.ru.itiscl:test-terget>
    backing-store /storage/lun0.img
    initiator-address 10.0.2.10
    incominguser user password10
</target>
```

- `iqn.2021-10.ru.itiscl:test-terget` - `iqn` + дата регистрации домена + название домена в обратном формате + `:` + уникальное имя таргета на данном компьютере.
- `backing-store /storage/lun0.img` - файл/устройство/диск, который будет экспортирован.
- `initiator-address 10.0.2.10` - ip адрес клиента, может быть несколько строчек.
- `incominguser user password10` - логин и пароль, которыми инициатор должен аутентифицироваться на таргете, можно убрать.

Таких таргетов можно описать множество, можно в одном конфигурационном файле, можно в разных.

`sudo service tgt restart` - перезапускаем сервис tgt.
`sudo service tgt status` - просмотр состояния сервиса tgt.
`sudo tgtadm --mode target --op show` - посмотрим, какие таргеты "опубликованы", то есть какие таргеты есть на этом сервере.

Клиентская часть

Инициатор

`sudo apt show open-iscsi` - вывод информации о пакете open-iscsi.
Видим, информацию ("Description: **iSCSI initiator tools**") о том, что это как раз инициатор клиентской части.
`sudo apt install open-iscsi` - устанавка пакета open-iscsi.
`dpkg -L open-iscsi | less` - показывает те файлы, которые установил пакет open-iscsi.
`sudo nano /etc/iscsi/iscsid.conf` - конфигурационный файл, который отвечает за поведение самого демона (клиент тоже работает всё время в фоновом режиме).

Отредактируем этот файл.

- Изменим параметр, отвечающий за режим запуска (startup settings):
  - ракомментируем строку `node-startup = manual`;
  - раскомментируем строку `node-startup = automatic`.
- Изменим параметр, отвечающий за метод аутентификации (CHAP settings):
  - раскомментируем строку `node.session.auth.authmethod = CHAP`, метод аутентификации для сессии;
  - раскомменитруем строки `node.session.auth.username = user` и `node.session.auth.password = password10`, устанавливающие логин и

пароль, чтобы они стали совпадать с логином и паролем, которые указаны в таргете;
- аналогично раскомментируем `discovery.sendtargets.auth.authmethod = CHAP`;
- аналогично раскомментируем `discovery.sendtargets.auth.username = user` и `discovery.sendtargets.auth.password = password10`.

`sudo service open-iscsi restart` - перезапускаем сервис open-iscsi.
`sudo service open-iscsi status` - просмотр состояния сервиса open-iscsi.

`sudo iscsiadm --mode discovery --portal 10.0.2.150 --type sendtargets` - подключаемся в порталу, находим там все таргеты и просим таргеты прислать нам информацию.

В /etc/iscsi появился подкаталог nodes.
`cd nodes/` - переходим и видим внутри него подкаталог iqn.2021-10.ru.itiscl:test-terget.
`cd iqn.2021-10.ru.itiscl:test-terget/` - переходим и видим внутри него подкаталог 10.0.2.150,3260,1.
`cd 10.0.2.150,3260,1/` - переходим и видим один единственный файл default.
`nano default` - открываем его на редактирование. В нём можем задавать конкретные настройки для этой ноды. Настройки по умолчанию берутся из iscsid.conf.

Теперь нужно залогиниться во все ноды, которые у нас настроены.
`sudo iscsiadm -m node --login` - подключиться ко всем таргетам.
`sudo iscsiadm -m node --targetname "iqn.2021-10.ru.itiscl:test-terget" --login` - залогиниться к конкретному таргету.

`lsblk` - просмотрим блочные устройства, которые есть в системе. Видим, что появляется ещё один диск sdb (тот самый удалённый сетевой диск размером в 2 ГБ). Теперь есть возможность разбить его на разделы, смонтировать, полноценно с ним работать.
`sudo fdisk /dev/sdb` - общее название системных утилит для управления разделами жёсткого диска. Например, можем разбить диск sdb на два раздела sdb1 и sdb2.
`mkfs.ext4 /dev/sdb1` - например, отформатируем раздел sdb1 в ext4.

Команды для того, чтобы разлогиниться.
`sudo iscsiadm -m node --logout` - отключиться от всех таргетов.
`sudo iscsiadm -m node --targetname "iqn.2021-10.ru.itiscl:test-terget" --logout` - разлогиниться от конкретного таргета.

Просматривая блочные устройства с помощью `lsblk`, убеждаемся, что удалённый сетевой диск отсутствует в списке.

Заключение

Эту технологию обычно используют для организации кластеров виртуальных машин. Один физический сервер можно выделить как scsi-target, и он будет раздавать блочные устройства как scsi. А система виртуализации может забирать эти диски для своих виртуальных машин как инициатор. И тогда, если есть один физический сервер, который хранит образы дисков и раздаёт их по iscsi-протоколу, и если мы, например, взяли две физические машины, на которых хостятся виртуальные машины (а на них вообще нет жёстких дисков), те жёсткие диски подцепляются к виртуальным машинам по iscsi-протоколу. Это даёт возможность мигрировать одну машину с одного физического хоста на другой. Сам жёсткий диск копировать не надо, потребуется только скопировать содержимое оперативной памяти и состояние этой виртуальной машины. И таким образом, виртуальные машины могут мигрировать из одного физического сервера в другой физический сервер, а их жёсткий диск находится всегда на хранилище, где scsi-target.