How can I mount a block device from one computer to another via the network as a

Is it possible to export a block device such as a DVD or CDROM and make it so that it's mountable on another computer as a block device?

NOTE: I'm not interested in doing this using NFS or Samba, I actually want the optical drive to show up as a optical drive on a remote computer.

/ linux / mount / block-device / dvd / data-cd

edited Mar 12 '14 at 23:27 — Gilles — 337k 57 599 1034

asked Mar 12 '14 at 16 — slm ♦ — 150k 34 2

4 Answers

I think you might be able to accomplish what you want using network block devices (NBD). Looking at the wikipedia page on the subject there is mention of a tool called `nbd` . It's comprised of a client and server component.

Example

In this scenario I'm setting up a CDROM on my Fedora 19 laptop (server) and I'm sharing it out to an Ubuntu 12.10 system (client).

installing

```
$ apt-cache search ^nbd-
nbd-client - Network Block Device protocol - client
nbd-server - Network Block Device protocol - server

$ sudo apt-get install nbd-server nbd-client
```

sharing a CD

Now back on the server (Fedodra 19) I do a similar thing using its package manager YUM. Once complete I pop a CD in and run this command to share it out as a block device:

```
$ sudo nbd-server 2000 /dev/sr0

** (process:29516): WARNING **: Specifying an export on the command line is deprecated.

** (process:29516): WARNING **: Please use a configuration file instead.
$
```

A quick check to see if it's running:

```
$ ps -eaf | grep nbd
root     29517     1  0 12:02 ?        00:00:00 nbd-server 2000 /dev/sr0
root     29519 29071  0 12:02 pts/6    00:00:00 grep --color=auto nbd
```

Mounting the CD

Now back on the Ubuntu client we need to connect to the `nbd-server` using `nbd-client` like so. **NOTE:** the name of the nbd-server is greeneggs in this example.

```
$ sudo nbd-client greeneggs 2000 /dev/nbd0
```

```
Negotiation: ..size = 643MB
bs=1024, sz=674983936 bytes
```

(On some systems - e.g. Fedora - one has to `modprobe nbd` first.)

We can confirm that there's now a block device on the Ubuntu system using `lsblk`:

```
$ sudo lsblk -l
NAME                MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                   8:0    0 465.8G  0 disk
sda1                  8:1    0   243M  0 part /boot
sda2                  8:2    0     1K  0 part
sda5                  8:5    0 465.5G  0 part
ubuntu-root (dm-0)  252:0    0 461.7G  0 lvm  /
ubuntu-swap_1 (dm-1) 252:1    0   3.8G  0 lvm  [SWAP]
sr0                  11:0    1 654.8M  0 rom
nbd0                 43:0    0   643M  1 disk
nbd0p1               43:1    0   643M  1 part
```

And now we mount it:

```
$ sudo mount /dev/nbd0p1 /mnt/
mount: block device /dev/nbd0p1 is write-protected, mounting read-only
$
```

did it work?
The suspense is killing me, and we have liftoff:

```
$ sudo ls /mnt/
EFI  GPL  isolinux  LiveOS
```

There's the contents of a LiveCD of CentOS that I mounted in the Fedora 19 laptop and was able to mount it as a block device of the network on Ubuntu.

edited Nov 8 '15 at 20:16 — maxschlepzig — **20.8k** 16 90 154

answered Mar 12 '14 at 16:33 — slm ♦ — **150k** 34 251 424

I get this: "WARNING Specifying an export on the command line is deprecated. Please use a configuration file instead". Version: 3.3 – guettli Aug 11 '14 at 13:41

@guettli - thanks for the feedback. What distro was this on? – slm ♦ Aug 11 '14 at 14:30

@sim openSUSE 13.1 – guettli Aug 11 '14 at 14:54

Note: `greeneggs` is the server's host name. – devius Nov 10 '14 at 16:34

1 Note that the old-style protocol (i.e. using one port for each device) was marked as deprecated for several versions and was dropped in nbd 3.10. – maxschlepzig Nov 8 '15 at 20:13

One alternative to `nbd` (if you're interested) is using iSCSI. `tgtd` can be configured to have a `/dev` device as its backing storage for a particular iSCSI IQN.

If you're on a RHEL system so you just need to install `scsi-target-utils` and then configure/start `tgtd` on the source system. Configuration of `tgtd` can get involved but Red Hat provides plenty of different examples for the various scenarios.

For Example:

```
<target iqn.2008-09.com.example:server.target4>
    direct-store /dev/sdb      # Becomes LUN 1
    direct-store /dev/sdc      # Becomes LUN 2
    direct-store /dev/sdd      # Becomes LUN 3
    write-cache off
    vendor_id MyCompany Inc.
</target>
```

To start it up on Fedora/RHEL:

```
# systemctl start tgtd.service
# firewall-cmd --add-service iscsi-target
```

You would install `iscsi-initiator-utils` on the client system and use `iscsiadm` to send targets

then to "log into" the enumerated targets. For Example:

```
# iscsiadm -m discovery -t sendtargets -p <remoteHost>
# iscsiadm -m node -T <Complete Target IQN> -l -p <remoteHost>
```

The iSCSI LUN's will then present to the system as regular block devices. On RHEL, you can check the transport a particular device is coming over you can just do an `ls -l /dev/disk/by-path | grep iscsi` to see what storage is coming over iSCSI. The path will also list the IQN of the target you logged into above.

When the iscsi device is not needed anymore one can remove it via:

```
# iscsiadm -m node -T <Complete Target IQN> -u -p <remoteHost>
```

edited Feb 25 at 21:30

answered Mar 12 '14 at 16:42

Bratchley
8,984 3 27 60

2 If you want you can add your A to this AU Q too, it's what prompted me to write this one up: askubuntu.com/questions/433231/… – slm ♦ Mar 12 '14 at 16:45

I would, but I don't really have an Ubuntu system handy to ensure the package names are the same or if the configuration examples I referenced are upstream or just something Red Hat does. The configuration itself should be the same, though. – Bratchley Mar 12 '14 at 16:54

1 Joel you're the best...no Ubuntu system 8-) – slm ♦ Mar 12 '14 at 17:05

1 Was actually easier to setup than NBD. The issue with current NBD is that most of the examples/documentation you find via google use the old-style configuration, where current versions removed the support for it. Also, version < 3.10 might have the read-only bug. – maxschlepzig Nov 8 '15 at 19:45

You obviously prefer the SAN solution. Beside the already mentioned iSCSI and NBD, you have also the AoE (ATA over ethernet) approach.

This is very easy to do:

On the serving side you need to

```
modprobe aoe
vbladed 0 0 eth0 /dev/sdc
```

On the client side

```
modprobe aoe
aoe-discover
aoe-stat
e0.0      1000.204GB       eth0 1024  up
```

Your devices are in

```
ls -l /dev/etherd/
c-w--w----  1 root disk 152, 3 Mar 12 22:47 discover
brw-rw----  1 root disk 152, 0 Mar 12 22:47 e0.0
brw-rw----  1 root disk 152, 1 Mar 12 22:47 e0.0p1
cr--r-----  1 root disk 152, 2 Mar 12 22:47 err
c-w--w----  1 root disk 152, 6 Mar 12 22:47 flush
c-w--w----  1 root disk 152, 4 Mar 12 22:47 interfaces
c-w--w----  1 root disk 152, 5 Mar 12 22:47 revalidate
```

Where `e0.0` is your `/dev/sdc` and `e0.0.p1` is `/dev/sdc1`

`dmesg` on server:

```
[221384.454447] aoe: AoE v85 initialised.
```

`dmesg` output on client:

```
[ 1923.225832] aoe: AoE v85 initialised.
[ 1923.226379] aoe: e0.0: setting 1024 byte data frames
[ 1923.226910] aoe: 38607725d8b1 e0.0 v4014 has 1953525168 sectors
[ 1923.653820]  etherd/e0.0: p1
```

Pretty easy.

Additional Notes

- `vbladed` is part of the package `vblade` on Fedora & Ubuntu, likely the same in other distros as well.
- `aoe-discover` & `aoe-stat` are part of the package `aoetools` on Fedora & Ubuntu as well.
- Device shows up in `fdisk` as a block device, for example, `/dev/etherd/e0.0` .
- The version of `vblade` that's available in the F19 and F20 repositories is pretty dated, it's version 14. The ATAoE project page has version 21 available. There's a updated RPM available for Fedora 19 x86_64 here.

edited Mar 26 '14 at 0:30 answered Mar 12 '14 at 21:50
user55518

Tried doing this b/w Fedora and Ubuntu, it's not working. – slm ♦ Mar 12 '14 at 22:23

I usually check everything before I post. Q: is aoe loaded? Did you give proper arguments? On receiver: did aoe found the opponent? What says `dmesg` ? – user55518 Mar 12 '14 at 22:27

I know you do, I wasn't implying that it was wrong only that there might be some extra steps 8-) – slm ♦ Mar 12 '14 at 22:30

1 ROTFL, and did you count the smartphone too? – user55518 Mar 12 '14 at 23:19

1 aaahh, well, thanks for the votes :) – user55518 Mar 12 '14 at 23:20

Although this solution works, it is presented only as a method to show that TMWTDI.

For this setup we need a unix inspired system to host the block device, and a system with a loop back block device for the system to view it. Linux will work for both.

Next we need a dumb network file system (it must not make optimizations of the data storage), like samba or webdav. Specifically NFS, and AFS will not work.

Place the block device in a folder that is exported. (`man mknod` will help.) disable block device checking if applicable. mount the exported filesystem on the client computer, create a loop device using the imported file as it's backing store and mount as normal. (`losetup` or `mount -o loop`)

How and why it works, and why the performance sucks: One of the core ideas behind unix is that everything is a file, so devices are files only they are a little special (that is why they are called special files). There are several types of special files that mknod can produce, but we are only going to look at block devices. block devices exist in the file system like any other file and like any other file they have owners, permissions, inodes and names; but they have two differences: they do not store their data in the filesystem, and they have extra capabilities so they can be mounted. If the file server is dumb enough not to pay attention to these differences, you can ignore them and export them as regular files. Next the loop device takes an ordinary file (local or remote) and adds the special properties and capabilities so that you can mount regular files as if they are block devices. The big drawback is performance.

answered Mar 14 '14 at 7:52

hildred
3,573 1 12 33