Clone an NTFS disk using Unix

Unix based rescue disks can be used to efficiently move an existing NTFS file system to a new disk using the common utility *ntfsclone*. *ntfsclone* only copies individual partitions, not the entire disk, so additional utilities will need to be used. Reasons for using this setup are:

- Much faster as ntfsclone only copies used space on the source disk
- Target disk can be any size equal to or greater than the sum of the source partitions
- “rescue mode” will ignore bad sectors, allowing partial file recover in some cases.

Note: With all these procedures, it is best to start with a clean file system. Run a scan disk (from inside Windows, or using *chkdsk C: /f /r* before you do anything, if possible.

Note: After any of these procedures, the NTFS partitions should be marked as *dirty*, meaning Windows will automatically perform a chkdsk on them. Do **not** bypass this; let Windows perform a disk scan and attempt to fix errors. If by some chance this is not initiated automatically, manually perform a scandisk on the file systems.

Tools Needed

You will need the following tools, generally available on all Linux distributions.

- **dd** Device Duplicate, available on all Unix based systems as far as I know
- **sfdisk** disk partitioner, can efficiently clone partition table from one disk to another. Available on Linux
- **ntfsclone** efficiently copy ntfs file system from one source partition to a target partition. Available on Linux and most Unix systems in the package ntfsprogs
- **gparted** *optional*. Needed if you want to resize partitions before or after copy.
- **kpartx** *optional*. Makes working with disk images (vs physical disks) easier.
- **pbzip2** *optional*. Highly compresses a file or bit stream using all available processors. If pbzip2 is not available, *bzip2* will work much slower (it is only single threaded).

All of these tools can be found on several “live” image distributions which can be burned to a CD or USB. Two of my favorites are systemrescuecd and Knoppix, but just about any live Linux CD will have them.

Procedure

using ntfs-clone

1. *optional* Resize/move original partition(s). This reduces the required size of the target disk, allowing you to downsize a machine, for example moving from a 500G hard drive to a 128G SSD.
2. Copy MBR from source to target (may not be needed, but I do it for safety)
3. Clone partition table from source to target

4. Copy file system from each source partition to target partition.
5. *optional* Resize target partitions to fill entire disk.

using dd to duplicate partition

1. *optional* mount partition(s) and write zero's to all unused sectors
2. use the dd command to copy the partition to a file, optionally using a compressor to make it smaller

Examples

Direct Disk Clone

Following example assumes your old disk is /dev/sda, with two partitions, /dev/sda1 and /dev/sda2. We have a new disk, /dev/sdb, which is the same size or larger than the sum of space used by /dev/sda1 and /dev/sda2.

Be Very Careful forgetting which disk is the target and which is the source can result in loss of data. Make backups, and read each command carefully before executing. This is especially important with the command ntfsclone, which is counter-intuitive.

```
# copy MBR, first 446 bytes in disk
dd if=/dev/sda of=/dev/sdb bs=446 count=1
# copy the partition table to the new device.
# New device MUST be same size or larger than original one
sfdisk -d /dev/sda | sfdisk sdb
# optional, set DMA on both drives to speed access
hdparm -d 1 /dev/sda
hdparm -d 1 /dev/sdb
# use ntfsclone to copy from /dev/sda1 to /dev/sdb1 in rescue mode
ntfsclone --rescue --force --overwrite /dev/sdb1 /dev/sda1
# copy second partition
ntfsclone --rescue --force --overwrite /dev/sdb2 /dev/sda2
```

- The first command simply copies the first 446 bytes of the source to the target disk; this is the Master Boot Record, and it might be set up already, but doesn't hurt anything and ensures we have a bootable device.
- The *sfdisk* command is only one way of copying a partition table. You can do the same thing using fdisk -l /dev/sda, copying the information, then using fdisk /dev/sdb and recreating it. You will need to make sure the start/end are set the exact same. sfdisk does it in one simple command, however.
- The *hdparm* commands simply ensure both drives can use Direct Memory Access, which greatly speeds reading and writing. Optional, but this can significantly reduce the time to make the partition copies.
- Finally, the two *ntfsclone* commands copy the partitions. Usually the first partition is a very small hidden NTFS which contains the boot information, so it happens very fast. **Be very careful** when putting the source and target partitions in as they are “reversed” from many commands. The target is part of the parameters (–overwrite *target*) so are very easy for people

use to other commands to get backwards.
- The other two parameters to *ntfsclone* are not needed normally. *–rescue* will not stop the copy if it encounters a read error on the source; it will fill the target with question marks. *–force* will perform the task even if the source file system is marked as dirty; without *–force* the program will abort and ask you to clean up the file system before continuing (good idea if you can do that).

Save to a compressed file

ntfsclone is also very good for making an efficient backup of an NTFS partition. Save the output to a file instead of a partition; it will create an output file only large enough to store the data on the partition. Adding compression will further reduce storage requirements

```
ntfsclone --output - /path/to/partition | pbzip2 -c > /path/to/image/file
```

Note that pbzip2 uses a nice threaded setup that will max out all of your processor cores, resulting in an amazing space reduction. In one test, a 500G partition image with 172G used space, including all programs and the operating system, was compressed into a 25 gigabyte file. Since ntfsclone "zero fills" unallocated sectors, compression gives you fantastic results.

Disk Images

Disk images can be a single file or an LVM2 logical partition or something that is not a physical device. In this case, you need to work with the partitions embedded in the image.

There are some older ways of directly accessing the partitions, but if you have the command `kpartx` (available on most Linux systems), it can be used to tell mapper to "break apart" the image and make an entry in /dev/mapper with the partitions.

```
# display what will happen first
kpartx -lv /path/to/lv/or/diskimage
# now, actually add the partitions to /dev/mapper
kpartx -av /path/to/lv/or/diskimage
######## do your work ##########
# unmap the partitions
kpartx -dv /path/to/lv/or/diskimage
```

In this case, kpartx -av will create the mapping and show you the resulting paths to the partitions. They are generally the same as the device itself, with the partition number at the end. You can now work with the partitions as if you were working with a disk partition; cloning and restoring.

Prior to making the copy, it is a good idea (though not a requirement) to zero out all of the unused parts of the hard drive. The following assumes you have an lvm block that is used by a Windows machine (Windows 7 to whatever) named 'win_disk' which contains two partitions. The first partition is generally small, and used for booting, and the second partition is generally large where Windows and any apps are installed.

```
kpartx -lv /dev/vg0/win_disk # see what it will do
```

```
kpartx -av /dev/vg0/win_disk # create the partition mapping
ntfs-3g /dev/mapper/vg0--win_disk2 /mnt # mount the second partition at /mnt
using ntfs
# there are two files in the root directory of the disk which are for
hibernation and swap space
# they can be safely removed if you did not hibernate the disk
rm /mnt/hiberfil
rm /mnt/swapfile # actually, I'm not sure what this filename is, so look and
see
rm -fR /mnt/Windows/Temp/* # potentially dangerous, but haven't had issues
yet
mkdir /mnt/freespace # a directory for our zero byte files
df -h # see how much free space we have, use that number in the following
command
# in the following command, change 20 to the amount of space you want to
zero out
for i in {01..20} ; do echo Loop $i ; dd if=/dev/zero
of=/mnt/freespace/deleteme.$i bs=1M count=1024 ; done
# remove the files created
rm -fR /mnt/freespace
# unmount the partition
umount /mnt
# delete the partition mapping
kpartx -dv /dev/vg0/win_disk
dd if=/dev/vg0/win_disk | pv -petrs sizeOfPartition | pbzip2 -c >
/output/file/name/that/has/space
```

Resizing

I use gparted to do my resizing because it is so simple. The only problem is that it is GUI based, so requires an X-Windows interface to run. *gparted* can extend, contract or move an NTFS partition without data loss. This is included in many live cd images, or you can download a live cd specifically for gparted.

That being said, here is how to manually resize without resorting to *gparted*, so no GUI.

```
# manually reduce /dev/sda2 to 100G.
# /dev/sda2 has less than 100G of data on it.
ntfsresize --size 100G /dev/sda2
fdisk /dev/sda2
# in fdisk, use the "p" option to display the current partition scheme
# then delete /dev/sda2
# Add /dev/sda2 starting point
# and extending it to 100G
```

The opposite, enlarging a partition, is done just backwards from reducing

```
fdisk /dev/sda
# use the 'p' command
```

```
# delete /dev/sda2
# recreate it from the same starting point
# and extending it out to whatever size you want
ntfsresize /dev/sdb
# with no size parameters, ntfsresize will grow
# to the size of the partition
```

While there are no known issues with either ntfsresize or fdisk, there is always the chance you'll mess up and trash something you didn't want to, so it is definitely good to make a backup.

You'll see why I like gparted! The authors have done a great job bringing together the ntfs-progs toolbox and partition tools. Makes it less likely to mess up.

References

- https://nilbus.com/linux/disk-copy.php
- http://gparted.sourceforge.net/ - gparted live CD
- http://www.system-rescue-cd.org/ - systemrescuecd bootable utility
- http://knoppix.net/ Knoppix bootable utility CD

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://kb.unixservertech.com/microsoft_windows/clone_disk_using_linux

Last update: **2019/01/29 08:05**