The Interoperability Power of Linux-NTFS Tools

Steven Mathes

Abstract

Some of the best Windows utilities available might be in your Linux partition.

A Linux environment can gain access to dozens of filesystems, whether on the local hard drive or somewhere on the network. More specifically, Linux can run many tools to manipulate Windows filesystems or repair Windows problems.

One suite of tools comes from the Linux-NTFS Project. These utilities work many miracles. One resizes NTFS partitions. Several manipulate individual files. One clones an entire NTFS image. It is possible to back up Windows installations, clone new workstations from a centrally stored image and update images across a network. And, because these tools run inside Linux, they benefit from the power of the Linux environment. These tools help when you're dealing with a single dual-boot computer. They quickly become indispensable if you work with a large network. Aided by redirection, pipes and scripting, it is easy to automate many tedious but important Windows maintenance tasks from within Linux.

Installation

The utilities are widely available and well supported. Packages are available for virtually all Linux distributions that have package managers, and the software itself is even included on the Knoppix live CD. Many distributions install the tools to be run only by the root user. To see if these tools are on your installation of Linux, consulting the man pages will at least show whether the documentation is installed: **`man ntfsprogs`**.

Even if the software and/or documentation are absent, you can install these tools yourself. For SUSE, Debian, Ubuntu and Gentoo, ntfsprogs is the package name to search for and install. The packages for some distributions include all of the NTFS tools, some do not. For example, the package in the Etch version of Debian includes the ntfsmount tool, and the package in the Sarge version does not. Red Hat/Fedora distributions do not support NTFS, based on perceived licensing issues, but specifically designed packages for Red Hat/Fedora are available directly from the Linux-NTFS Project. Of course, consulting the actual home page of the project (http://www.linux-ntfs.org) gives the most up-to-date documentation and information, as well as the latest source code and instructions for building the complete set of tools.

No matter what flavor of Linux you run, it is possible to download the source code and install from that. This is a good choice if you want the newest features and the latest NTFS drivers, although you could suffer from the disadvantage of having bypassed your package manager.

Note: before you build ntfsprogs from scratch, you probably should install the FUSE library (http://fuse.sourceforge.net). Linux has a built-in NTFS driver, but the NTFS utilities include a second driver for NT filesystems. The non-native driver is the FUSE-based ntfsmount, which boasts many extra features. However, it is a bit slower than the driver that comes with the latest kernel. Furthermore, it requires that your kernel has the FUSE module.

If you want to install the FUSE library, download the latest source and store it in a handy directory, maybe the same place you plan to store your ntfsprogs download. The installation follows the “configure, make, make-install” process that has become the standard (note that the version number may have changed by the time you read this). Do this as root:

```
tar -xzvf fuse-2.5.3.tar.gz
cd fuse-2.5.3.tar.gz
./configure
make
make install
```

Installing the FUSE library and module is not completely necessary if all you want is read access (and somewhat temperamental read/write access) to an NT filesystem. That's because for all distributions, except Red Hat/Fedora, there is a native Linux kernel driver that runs through the normal mount command. It is faster, but it lacks the extensive features and feedback of ntfsmount.

Now, download the ntfsprogs source, and then save it in a handy directory. Operating as root, build it much the way you built the FUSE package (again, the actual version number may have changed by the time you read this):

```
tar -xzvf ntfsprogs-1.13.1.tar.gz
cd ntfsprogs-1.13.1
./configure
make
make install
```

When building ntfsprogs without the FUSE library (even if you do have the FUSE module), you will get a complaint while running the configure command:

```
checking for FUSE_MODULE... configure: WARNING: \
ntfsmount requires FUSE version >= 2.3.0
```

This shouldn't be fatal to building the other NTFS tools, but you will not be able to compile ntfsmount.

If you are running Red Hat/Fedora, you might not even have the kernel driver. In that case, it is strongly recommended that you either install a custom kernel containing the kernel-based NTFS driver or install the FUSE libraries before building.

The Software

At this point, it is assumed that you have either installed ntfsprogs or have discovered it already installed on your system.

If you have already looked at the ntfsprogs man page, you have seen the complete list of the utilities. Here is that part of the output from the man page:

```
mkntfs(8) - Format a partition using NTFS.
```

```
ntfscat(8) - Dump a file's contents to the standard
output.
ntfsclone(8) - Efficiently clone, create, restore or
rescue an image of an NTFS partition.
ntfscluster(8) - Locate the owner of any given sector
or cluster on an NTFS partition.
ntfscp(8) - Overwrite file on an NTFS partition.
ntfsfix(8) - Check and fix some common errors, clear
the LogFile and make Windows perform a thorough check
next time it boots.
ntfsinfo(8) - Show some information about an NTFS
partition or one of the files or directories
within it.
ntfslabel(8) - Show, or set, an NTFS partition's
volume label.
ntfsls(8) - List information about files in a
directory residing on an NTFS partition.
ntfsmount(8) - NTFS module for FUSE.
ntfsresize(8) - Resize an NTFS partition without
losing data.
ntfsundelete(8) - Recover deleted files from an
NTFS partition.
```

Many of the utilities listed are more useful to developers than to people doing maintenance on a network or dual-boot computer. However, some of these are real life-savers, and ntfsclone is the biggest life-saver of all.

Using the NTFS Tools

In order to try out ntfsclone, you need a computer with the NT filesystem to clone, and it needs to have access to another filesystem large enough to store the image. Recommended filesystems are ext2, ext3, xfs or ReiserFS. The documentation for ntfsclone warns that the ReiserFS is slow when handling sparse files, but I have found the performance to be okay with more recent versions. It is possible to to use an external drive, as long as it has the ability to store huge files -- for some operations you will need space as large as your entire Windows partition. If you have an external drive formatted as a FAT32 filesystem, it will have a size limit for individual files that is too small for what you need. Of course, if your computer does not have Linux installed, you will need to boot from a live Linux CD, such as Knoppix.

Notice that the description of the ntfsclone utility above claims that it does its job “efficiently”. This is not merely a boast. On newer hardware, it can clone a substantial Windows XP workstation in just a couple of minutes. If you had an NT filesystem on the first partition of the first IDE drive and were operating from Linux on the same computer, the following command would back up the NTFS as a single file:

```
ntfsclone /dev/hda1 -O /usr/local/backup/ntfs.img
```

The uppercase O in this command tells the software to overwrite the image, but it will create the file if it is absent. This will not compress the filesystem. In fact, it will leave it in a state to allow you to mount ntfs.img using loopback. First, make a mountpoint:

```
mkdir /usr/local/backup/mtpt
```

Then, use ntfsmount and the same syntax you would use for an ordinary mount:

```
ntfsmount -o loop /usr/local/backup/ntfs.img \
 /usr/local/backup/mtpt/
```

The ntfsmount command mounts the filesystem read/write by default. Files can be copied, moved and deleted easily. Of course, there are the usual cross-platform perils to contend with. For example, situations involving configuration files can require caution when alien line endings and character sets are involved.

Using the the native mount command with the native driver involves the same familiar syntax:

```
mount -o rw,loop,nls=utf8 -t ntfs \
 /usr/local/backup/ntfs.img \
 /usr/local/backup/mtpt/
```

Note that this mount also makes a provision for a Windows-compatible character set. You still need to use caution, finesse and expertise, however, if you were to choose to edit, say, boot.ini with Emacs. It would be better to edit such a file in a Windows environment or perhaps with Notepad running through Wine.

If you want read/write access, your success with this last mounting method might vary according to the version of your kernel. Again, the native driver is a bit finicky. It may complain, and if it does, its usual behavior is to fall back to a mount that is read-only. Older versions of the native driver are outright dangerous in read/write mode.

Unmount the filesystem the same way for both methods. From the directory containing the mountpoint do the following:

```
umount mtpt/
```

The ntfs.img file can be moved and copied just like any other (admittedly huge) file. It can be compressed and stored in a safe place. It can be uploaded to remote locations. A copy can be edited and then restored over the original. The command for restoring this backup onto the original partition (while in the directory containing the backup) is as follows:

```
ntfsclone  ntfs.img -O /dev/hda1
```

Sometimes, smaller is better. The ntfsclone command will take flags that allow your image to be compressed efficiently. These flags also make the process of cloning much faster, both from the local hard drive and over the network. Here is one example, where the image is saved much the way it was in the first example:

```
ntfsclone --save-image /dev/hda1 -O \
 /usr/local/backup/ntfs.img
```

This image, alas, cannot be mounted unless it is restored, either to its original partition or to a different file. Restoring to its original partition would happen as follows:

```
ntfsclone --restore-image --overwrite /dev/hda1 \
/usr/local/backup/ntfs.img
```

Note that in the above, the -O has been replaced by the more script-friendly --overwrite flag. They do the same job. All flags can be expressed as script-friendly words (for readers of English), and most can be expressed as single letters.

Now comes the good part. The ntfsclone utility will send its data to standard output. This means you have your choice of various compression utilities, different modes of transfer over the network and so forth. Any useful tool that accepts standard input could process the image. Here are some examples.

To back up a compressed image, do:

```
ntfsclone --save-image --output - /dev/hda1 | gzip \
 -c >ntfs.img.gz
```

The image is sent to standard output by the -output flag with the argument of a single dash. The gzip utility compresses it, then redirects the stream to overwrite or create the file ntfs.img.gz.

To back up the image to a remote computer, do:

```
ntfsclone --save-image -o - /dev/hda1 | ssh \
backups@storage.mydomain.org \
"dd of/home/backups/windows/images/ntfs.img"
```

Here, the flag for --output is shortened to its single-letter abbreviation. It is sent to standard output. This, in turn, is piped into the ssh program. The stream is sent over the network to a computer named storage under the care of a user named backups and stored in its proper place through the dd command.

Here is another example:

```
wget ftp://storage.mydomain.org/home/backups/
↪windows/images/ntfs.img.gz \
-O - |  gunzip | tee /usr/local/backup/ntfs.img | \
ntfsclone --restore-image --overwrite /dev/hda1 -
```

This could be a line taken directly from a cloning script, because it needs no password or other user input. It uses wget to download the compressed image, uses gunzip to unzip it, and then splits the data stream with the tee command, so that a backup copy of the image is stored in the Linux partition at the same time that it is redirected to the NT partition on /dev/hda1. This assumes that storage.mydomain.org has a functioning anonymous FTP dæmon. Other possible ways of downloading without user input would be to use wget with Apache or to set up encryption keys to use with SSH. Again the possibilities are limited only by the incredible number of tools available.

Another useful tool in the ntfsprogs package is ntfsresize. This does exactly what it advertises. It shrinks or expands an NT filesystem. It operates on filesystems occupying partitions, but it also resizes filesystems that have been stored as single files by ntfsclone.

Note that ntfsresize doesn't change partition tables, it changes only the NT filesystem inside the partition. Changing the partition table is a job for fdisk or sfdisk.

This article does not cover how to partition a disk. A detailed and cautious description of how to free space on a drive occupied entirely by a single NT filesystem could take an article at least as long as this one. The operation itself doesn't take long, but it is a bit dangerous. Carelessness, or even bad luck, could result in a computer that refuses to boot. Given this, and given that the workaround of an extra hard drive costs almost the same as a tank of gas, this article continues to assume that partitioning already has been done.

Suppose, however, that the NT partition is just a little too small for the NT filesystem. This can happen, for example, if you don't account for the need of most partitioning tools to round down to a nearby sector, or if you replaced a defective drive with one having the same advertised size but with a different geometry.

The ntfsclone utility will work just fine on a partition that is too big, but it refuses to fit into a space that is even the slightest bit too small.

In that case, the ntfsresize tool can come to the rescue. To figure out how much space you could shrink out of your NT filesystem, type the command that follows (from the directory containing ntfs.img):

```
ntfsresize --info ntfs.img
```

The software will report something like the following:

```
ntfsresize v1.11.2
Device name        : ntfs.img
NTFS volume version: 3.1
Cluster size       : 4096 bytes
Current volume size: 90009203200 bytes (90010 MB)
Current device size: 90009203200 bytes (90010 MB)
Checking filesystem consistency ...
100.00 percent completed
Accounting clusters ...
Space in use       : 6508 MB (7.2%)
Collecting resizing constraints ...
You might resize at 6507421696 bytes or 6508 MB
(freeing 83502 MB). Please make a test run using both
the -n and -s options before real resizing!
```

This reports that you could shrink your filesystem down to as little as 6,508MB. Windows probably wouldn't run if you reduced it to the minimum size; it would be smart to leave a little room for future growth anyway. Note that the software advises that you could make a “test run using both the -n and -s options”. Instead, you simply could keep a backup copy in a safe place in case something goes wrong. Or, you could do both. Shrinking the filesystem to 10,000MB requires the following command:

```
ntfsresize --size=10000M ntfs.img
```

This produces a great deal of feedback, including the following:

```
100.00 percent completed
Updating $BadClust file ...
Updating $Bitmap file ...
Updating Boot record ...
Syncing device ...
Successfully resized NTFS on device 'ntfs.img'.
```

This should create an NT filesystem small enough to fit into its designated partition.

Conclusion

The NTFS tools may not be a requirement for everyone wanting a secure Windows workstation, but they do make life a lot easier.

In the context of a single dual-boot computer, complete backups can be performed to a safe, non-NTFS partition, either on the same hard drive, or even onto a removable hard drive of sufficient capacity. This may not make the effort worthwhile for everyone. However, for the user already equipped with a dual-boot system, the tools for greatly enhanced security may already be installed.

For a network administrator in charge of many Windows workstations, the potential is even greater. Dual-boot computers can be equipped with a shared disk partition (see Kevin Farnham's article “The Ultimate Linux/Windows System” in the June 2006 issue of *Linux Journal*). If GRUB is installed in this shared partition, along with alternate menu files, scripts can be written that reboot the computer into runlevels that automatically restore the Windows image, update it and so on.

Windows and Linux may be competitors in many areas. However, one of the great strengths of Linux is its open nature and the versatility of its command-line tools. The Linux-NTFS tools open up a conversation with the NT filesystem that, because of its one-way nature, makes for ideal security.