

Communicating With the Other Half: NTFS Support in Linux

By Roderick W. Smith

NTFS support has been slow in coming to Linux. But new drivers are now available that enable you to read from, and even to write to, NTFS partitions, including removable media formatted with NTFS. We show you how.

More than twenty years ago, Microsoft licensed its operating system to IBM for inclusion with its new personal computer. *PC-DOS*, as it was known in its IBM form, had many features and limitations, but one feature that has grown in importance well beyond the realm of the DOS world is its filesystem. The *File Allocation Table* (FAT) filesystem, named after its key data structure, is perhaps the most widely implemented filesystem in the history of computing.

The FAT file system is used (or at least supported) by everything from digital cameras to mainframe computers. This makes FAT the filesystem of choice for cross-platform data exchange on removable disks, and also an excellent way to exchange data between OSes in a multi-boot configuration on a single computer.

FAT, however, has its problems. For this reason, when Microsoft released its New Technology (NT) version of Windows, the software giant introduced a new filesystem, the *New Technology File System* (NTFS). Every version of Windows derived from Windows NT, including Windows XP and Windows Vista, supports NTFS, and in fact NTFS is the preferred filesystem for most functions in these modern versions of Windows.

Unfortunately, Microsoft has kept many of the internal details of NTFS secret, and the filesystem has changed with each new version of Windows. The secrecy makes it difficult for third parties, and particularly for open source developers, to write reliable NTFS drivers. In the case of a file system, an unreliable driver can translate into lost data, so NTFS support (particularly read/write NTFS support) has been slow in coming to Linux.

Fortunately, this picture is beginning to change, with several NTFS drivers now available for Linux. These drivers enable you to read from, and even to write to, NTFS partitions, including removable media formatted with NTFS. If you dual-boot between Linux and Windows, this will enable you to eliminate any FAT partition you might otherwise use for data exchange. Even if you don't let Windows near your hard disks, NTFS support will enable you to read removable disks that others have formatted with NTFS.

NTFS Options for Linux

Several tools exist that support NTFS in Linux:

- Linux kernel driver The Linux kernel has long included an NTFS driver. In the distant past, this driver supported a reliable read-only mode and a rather unreliable write mode. More recent versions (included in 2.5.11 and later kernels) have been extensively overhauled, and now provide reliable read support and reliable— but limited— write support. In particular, many directory options aren't supported.
- ntfsprogs This project uses FUSE to provide access to NTFS volumes. (For more on FUSE, see "Lighting the FUSE") The ntfsmount program mounts NTFS volumes with more features than the standard kernel driver, but because it's a userspace driver, it's slower. (Many distributions' binary packages omit the ntfsmount utility in favor of the NTFS-3G driver.) Although a 2.0.0 release is available, I've seen claims of unreliability in this version, so you may want to stick with a 1.1.x

Communicating With the Other Half: NTFS Support in Linux

By Roderick W. Smith

release (1.1.3 is current as I write) until the 2.x line stabilizes. The package as a whole includes utilities to create, resize, clone, and perform other actions on NTFS volumes. Utilities to directly access an NTFS volume, without mounting it, are also part of the package.

- **NTFS-3G** This package is a FUSE-based NTFS driver. It's a fork of an earlier version of `ntfsmount`, but it seems to be the favored tool of many distributions. It provides no ancillary utilities; for that, you should install `ntfsprogs`. The developers claim that its speed is comparable to that of many Linux-native filesystems, despite being a userspace driver. Check the website for more details.
- **Paragon** Paragon Software Group has developed a commercial NTFS driver for Linux. This driver provides full read/write support, but it's a binary driver without source code.
- **Captive** This project takes an unusual approach to NTFS support: It uses an open source wrapper and the `NTFS.DLL` driver file from Windows itself to provide NTFS access. This means that the hard work of parsing the NTFS data structures is done by Microsoft's own code. This sounds great, but the implementation (via FUSE) tends to be slow. The project has also been abandoned, so although you might get `Captive` working, it might stop working one day.

That's the overview. The rest of this column describes the NTFS-3G driver and `ntfsprogs` utilities in more detail. Although the other Linux NTFS tools may be useful in particular situations, these two packages are the most capable and popular ones available today.

Using NTFS-3G to Access NTFS Volumes

Most modern Linux distributions ship with NTFS-3G, available in a package called `ntfs3g` or `ntfs-3g`. Thus, you should be able to install the NTFS-3G driver by using `APT`, `yum`, `emerge`, or some other package management tool. When you do so, necessary dependencies, such as `FUSE`, will be installed along with the NTFS-3G driver itself, simplifying the process.

If your distribution lacks explicit NTFS-3G support, you can download the source code from the project's Web page. As I write, the latest stable version is 1.1004. The build and installation process is a conventional one: Extract the source code, change to the source code directory, type `./configure` to configure the package, type `make` to build the software, and type `make install` as `root` to install the software. Because NTFS-3G relies on `FUSE`, though, you may need to install it before you can complete your NTFS-3G installation.

Once NTFS-3G is installed, accessing your NTFS volumes is a matter of mounting them with a filesystem type code of `ntfs-3g`:

```
#mount-t ntfs-3g /dev/hda2 /mnt/windows
```

This command mounts `/dev/hda2` at `/mnt/windows` using the NTFS-3G driver. As just specified, though, the command is rather limited; NTFS-3G defaults to giving ownership of all files to the user who mounted the filesystem, with permissions of `0777` on all files, so every user will be able to read, write, and (theoretically) execute every file. To change permissions and ownership, you must include additional options, similar to those used to achieve the same effects with `FAT` or most other non-Unix/Linux filesystems:

```
#mount-t ntfs-3g-o uid=523,umask=0013 /dev/hda2 /mnt/windows
```

Communicating With the Other Half: NTFS Support in Linux

By Roderick W. Smith

This example gives ownership of all files to whoever has user ID 523, with permissions of 0764. You might prefer to use `fmask` and `dmask` rather than `umask`; the former two options set file and directory masks independently, enabling you to give directories execute permissions (as is normal in Linux) while keeping those permissions off of ordinary files (since you probably won't store Linux executables on an NTFS partition).

If you want to have your system mount NTFS partitions automatically on startup, or give ordinary users the ability to mount and unmount NTFS volumes, you can add an entry to `/etc/fstab` for this purpose. This entry will look like any other `/etc/fstab` entry, except of course for the fact that it will specify a filesystem type of `ntfs-3g`:

```
/dev/hda2 /mnt/windows ntfs-3g users,uid=523,noauto 0 0
```

If ordinary users should be able to mount and unmount the partition (as implied by the `users` option in the preceding line), you may need to make one additional change to your configuration: The NTFS-3G mount helper program (typically installed as `/bin/ntfs-3g`, `/sbin/mount.ntfs-3g`, or something similar) must be set SUID `root`. If this is not done, then only `root` will be able to mount and unmount NTFS volumes, even if the `users` or similar options are specified in `/etc/fstab`. To make this change, use the `chmod` utility:

```
#chmod a+s /bin/ntfs-3g
```

You should set the SUID bit on the file that's installed on your system, of course; it may or may not be called `/bin/ntfs-3g`. Use your distribution's package manager or simply look for files with `ntfs` in their names in the `/bin`, `/sbin`, and perhaps `/usr/bin` and `/usr/sbin` directories, or in the `/usr/local` equivalents if you installed from source code.

With these procedures in place, you should now be able to access files on your NTFS partitions, as well as on removable NTFS media. You should be able to read and write files, create new files, and otherwise treat the filesystem as if it were a FAT filesystem. Note that the NTFS-3G driver doesn't yet support Linux-style ownership and permissions, except insofar as you set them globally in mount options. Thus, you can't really use an NTFS volume as a fully functional substitute for a Linux-native filesystem such as Ext3 or ReiserFS. You can, though, copy files on and off of such filesystems for interchange with another OS on the same computer or use on another computer.

Manipulating NTFS Volumes

Although the NTFS-3G driver permits access to your NTFS volumes, you'll need other tools if you need to manipulate these volumes in any way. Such manipulation can be handy if a disk has become corrupt, if you need to resize a volume, or if you need to create a new NTFS volume, to name just three examples.

The `ntfsprogs` package delivers the tools you'll need to help out in such situations. Use your package manager to check the files installed from this package, or review the programs and documentation that are built from a source package, to learn precisely what this package includes. I'll present a brief run-through demonstrating several common NTFS operations.

To begin, you should have a test partition. I use `/dev/sdb4` (a Zip disk on my system) in the following examples. The `mkntfs` program (often also accessible as `mkfs.ntfs`) creates an NTFS volume:

Communicating With the Other Half: NTFS Support in Linux

By Roderick W. Smith

```
#mkntfs /dev/sdb4
```

This process can take a while; mkntfs is much slower than most other Linux filesystem-creation tools. Consult the man page for mkntfs to learn about its many options. One option you might want to use is -L name, which lets you set the volume name. If you want to change the volume name, or if you forget to set it with mkntfs, you can set it with the ntfslabel command:

```
#ntfslabel /dev/sdb4 NTFS_test
```

At this point, the filesystem is ready for use, so it can be mounted and accessed as described earlier. Many subsequent NTFS utilities are pointless without files on the filesystem, so I mounted it, copied some files, unmounted the filesystem, mounted it again, deleted one file, and then unmounted the volume:

```
# mount -t ntfs-3g /dev/sdb4 /mnt/zip
# cp * /mnt/zip
# umount /mnt/zip
# mount -t ntfs-3g /dev/sdb4 /mnt/zip
# rm /mnt/zip/file4.txt
# umount /mnt/zip
```

A series of programs, ntfsls, ntfsstat, and ntfsck, work on *unmounted* NTFS volumes in ways similar to the standard Linux ls, cat, and cp commands. The ntfsck command is very limited, though; it only enables you to overwrite existing files on an NTFS volume, not copy files off an NTFS volume or copy a file from Linux to a new NTFS file. A few commands demonstrate these utilities:

```
# ntfsls /dev/sdb4file1.txtfile2.txtfile3.txt# ntfsstat /dev/sdb4 file1.txt >
file1.txt.bak# ntfsck /dev/sdb4 file1.txt file3.txt
```

Note that this example uses ntfsstat as a stand-in for an NTFS-to-Linux cp command, by redirecting output to a normal file. You can of course use it to view the contents of a text file instead. The ntfsck command copies the local *file1.txt* to overwrite file3.txt on the NTFS volume. You can verify that this operation occurred as expected by copying the file back with ntfsstat or by mounting the volume using any Linux NTFS driver.

Earlier, I copied a file to the NTFS partition and then deleted it. I did this to enable demonstration of the ntfsundelete tool, whose function you can ascertain from its name. This tool is complex, so you should consult its man page for details; however, a basic demonstration can be fairly simple:

```
# ntfsundelete --scan /dev/sdb4>
Inode  Flags  %age  Date          Size  Filename
-----
16     F...    0%    2007-10-18    0
17     F...    0%    2007-10-18    0
18     F...    0%    2007-10-18    0
19     F...    0%    2007-10-18    0
20     F...    0%    2007-10-18    0
21     F...    0%    2007-10-18    0
22     F...    0%    2007-10-18    0
23     F...    0%    2007-10-18    0
30     FN..    100%  2007-10-18  1144
```

Communicating With the Other Half: NTFS Support in Linux

By Roderick W. Smith

Files with potentially recoverable content: 1

```
# ntfsundelete -undelete -inode 30 -output recovered.txt /dev/sdb4
```

These commands create a file (*recovered.txt*) on the local Linux filesystem from the file with inode number 30 identified in the scan operation. You may also be able to specify filenames using the `--match` pattern option rather than the `--inode` number option; however, filenames are often lost even when the files still exist. Of course, as with any undelete operation on any filesystem, recovery isn't guaranteed; the space allocated to a file may have been overwritten between the file's deletion and the recovery attempt. Even if a file is created, it could be truncated or include garbage not present in the original file.

Another useful maintenance tool is `ntfsclone`, which lets you create a backup of an NTFS volume. The `--save-image` (`-s`) option lets you create an image file backup that omits unused sectors for a space-saving backup, specifying the destination with the `--output` (`-o`) option:

```
#ntfsclone-s-o ntfs-backup.img /dev/sdb4
```

To restore the backup, you'll use the `--restore-image` (`-r`) option, but you must replace the `-o` option with the `--overwrite` (`-O`) option:

```
#ntfsclone-r-O /dev/sdb4 ntfs-backup.img
```

Note that you always specify the source filesystem *last* with `ntfsclone` and specify the target filesystem with the `-o` or `-O` option; you don't list the source and target in order, as you do with `cp`.

A few additional tools, such as `ntfsinfo`, `ntfscmp`, and `ntfsresize`, exist. However, the first two of these are most useful to NTFS developers. `ntfsresize` is best avoided in favor of `parted`, `QTParted`, or some other dedicated partition-resizing tool, since `ntfsresize` requires separate resizing of the filesystem and the partition that contains it, which can lead to errors if you're not careful.

Most of these utilities should only be used on unmounted partitions; using them on partitions that are mounted could cause confusion to whatever Linux NTFS driver you're using, particularly if the utility changes the filesystem in any way. In most cases, you must be *root* to use these tools, since ordinary users can't act on the Linux device files for partitions.