

Linux ntfsprogs

By Barry Shilliday

Many people coming to Linux for the first time will be dual-booting with a Windows XP installation on their hard drive. Those who use Linux day-to-day might still want to boot into Windows for a specific application they need. For these people some compatibility with NTFS, one of the filesystems Windows uses, is particularly useful.

For many years Linux has supported full read-only support for NTFS. Unlike open filesystems such as ext3 and Reiserfs, NTFS is closed and proprietary and has had to be reverse-engineered. This has meant limited write support, which until relatively recently has been considered too risky for any use besides testing and development. The good news is that things have moved on a lot.

The **Linux NTFS project** provides a set of tools called 'ntfsprogs'. The suite includes a tool to clone an NTFS partition, another to check consistency and fix problems, and another to resize an NTFS filesystem. All these tools are completely safe and have been used frequently by many Linux distributions. For example, when you install Linux and have the chance to free up some space from a Windows installation by resizing the partition, behind the scenes it is these tools that perform the checks and the resize.

A later addition to these tools is 'ntfsmount', a command that allows you to mount, and therefore access, an NTFS filesystem under Linux. This means there are two methods to mount an NTFS filesystem: the traditional kernel driver, which has been around for years; and the new userspace driver included in ntfsmount. The userspace driver is more up to date than the kernel driver and offers some safe support for writing as well as reading. As the name implies, the userspace driver does not need a driver in the kernel; it uses the 'fuse' or Filesystem in Userspace system instead.

Obtaining ntfsprogs

Most Linux distributions include ntfsprogs themselves. If you're using Ubuntu 6.10 ('Edgy Eft') you can install a recent copy with 'sudo apt-get install ntfsprogs' or load the Synaptic GUI for the point-and-click method.

Few, if any, distributions will have the latest version, which might have some big performance or feature enhancements, as well as the usual bug fixes. To compile and install the latest version, head over to the project's website and download the source code. Regular readers will be familiar with what comes next, the extraction, build and install:

```
# tar xfvz ntfsprogs.tar.gz
# cd ntfsprogs
# ./configure
# make install
```

Ensure the fuse development libraries are installed or 'ntfsmount' will not be built. On Edgy you can install these with 'sudo apt-get install libfuse-dev'. For ntfsmount to work you will also need the fuse-utils package.

Once installed you can check everything works by using the ntfsresize command with the -i option. This checks and reports on the filesystem without making any changes. To check the filesystem on /dev/hda2 run:

```
$ sudo ntfsresize -i /dev/hda2
```

Mounting NTFS

If you want to use the kernel driver or you do not have ntfsprogs, you can mount an NTFS partition using the normal mount command:

Linux ntfsprogs

By Barry Shilliday

```
$ sudo mount /dev/hda2 /mnt/xp
```

As above, this provides read-only support and makes all files readable by root only. To allow read-only access to anyone, add '-o umask=0' to the mount command.

To use the more recent driver and get some safe writing support, use ntfsmount instead. Some distributions may need the 'fuse' kernel module loading first; run 'modprobe fuse' as root to do this. This step isn't necessary with Ubuntu Dapper or Edgy versions. The ntfsmount command is almost identical to the standard mount:

```
$ sudo ntfsmount /dev/hda2 /mnt/xp
```

Since the filesystem is mounted as read-write, use the '-o uid=xxx' option to mount under your own id; you can run the 'id' command to find your own numeric id. On most systems the first created user has an id of 1000.

The ntfsmount command offers some safe writing support but it is far from complete. Modifying an existing file will always work; creating and deleting files can sometimes fail. When a failure occurs it is because the driver cannot guarantee safety; no harm is ever done to the filesystem.

These restrictions do certainly limit the usefulness of the driver, but if safety is required above all, it is still a good step forward. As with the kernel driver, there is never a problem with reading files.

Should you want Linux to mount an NTFS filesystem automatically at boot time, add an entry to the file /etc/fstab. Be careful editing this file as any damage could cause the system not to boot correctly. The dmask and fmask options allow read and write to all users; use the uid option as shown above with the mount command in order to restrict access to one user.

NTFS

3g

In July 2006 a new version of the NTFS driver was released as beta code. This is known as ntfs-3g, and includes a substantial rewrite of the code. The main change is full write support for regular NTFS filesystems, although at the time of writing this excludes compressed or encrypted files. The code has been thoroughly tested by thousands of people but, like all beta code, should be used with caution.

By the time you read this the final stable version may well have been released, in which case it should be safe to use freely. Ubuntu Edgy users can install a recent version of the code by running 'sudo apt-get install ntfs-3g'; this will also pull in the required fuse utilities.

To build the latest version from source, download and extract in the same way as for ntfsprogs. As with the ntfsmount tool, the fuse development libraries are required to build ntfs-3g.

Ntfs-3g is a direct replacement for 'ntfsmount' and will eventually replace it in the ntfsprogs suite. The command options are the same, simply use ntfs-3g instead:

```
$ sudo ntfs-3g mount /dev/hda2 /mnt/xp
```

You can use ntfs-3g in /etc/fstab in almost the same way as before; use the same options as shown in fstab.png, but replace 'ntfs-fuse' with 'ntfs-3g'. While ntfsmount gives access only to the root user by default, ntfs-3g gives access to all users, so bear this in mind if other people access your computer.

Linux ntfsprogs

By Barry Shilliday

A few months back the **Linux NTFS project** released beta drivers for full read-and-write access to NTFS partitions. Previously, read-only support was offered in the kernel, with write support considered unstable and for developers only.

The great news is that in February the new drivers, known as ntfs-3g, became stable and version 1.0 was released. This finally removes the need to have a Fat32 partition for people who want to exchange data directly between their Windows and Linux installations.

While adequate for small devices, Fat32 is not resilient enough for a modern large filesystem. Head over to www.ntfs-3g.org for more details on the new driver.

NTFS cloning

If you have installed Windows XP before, you will be familiar with the tedium involved: a long installation procedure, many driver installations to get the most basic hardware operating, and frequent reboots in between.

If you find yourself having to reinstall Windows a lot, all of this can be avoided by cloning the partition in Linux once the installation is finished. As with the standard 'dd' command, this makes an exact copy of the partition that can be saved elsewhere on the disk, and later placed back to restore the partition to the same state.

The Linux NTFS project provides a utility called ntfsclone, which performs this task for you. The difference with dd is that ntfsclone understands the filesystem and, as a result, it doesn't need to duplicate the entire partition. It needs only to save the used parts.

On a fresh installation this is particularly effective. If you have created a 60GB partition for Windows, the fresh install is probably using no more than about 3GB of that. Using dd would mean duplicating the entire 60GB somewhere else, while this tool requires only that 3GB of spare space.

You can get hold of the latest version of ntfsclone from the Linux NTFS website. It is part of ntfsprogs, an open-source suite of tools to manipulate the NTFS filesystem. In Ubuntu you can obtain a recent version by running `sudo apt-get install ntfsprogs`; unless you are running Feisty Fawn it is unlikely to be the latest. An alternative is to download the latest source code and build and install it yourself. You can do this with the usual 'configure and make' procedure.

The command is very simple to use. Specify the output file after -o (or `—overwrite` if it already exists) and provide the device name for the NTFS filesystem; for example, if `/dev/hda2` is the device and `ntfs.img` is the file we want to create:

```
$ sudo ntfsclone -o ntfs.img/dev/hda2
```

Any raw access to a partition requires root privileges; hence, sudo is used above. See the accompanying picture for the command in action, this time running as the root user. Here we can see that the partition size is 20,013MB (roughly 20GB), but only 4,575MB is in use.

After creating the image, using the file command on it reveals that it is a bootable NTFS image, just as would be produced with dd. However, using the du (disk usage) command shows that just 4.3GB of space is taken up with the image file.

To write this image back to the partition, simply reverse the options using `overwrite` instead, since the target file (the partition itself) already exists:

Linux ntfsprogs

By Barry Shilliday

```
$ sudo ntfsclone --overwrite/dev/hda2 ntfs.img
```

This copies the clone back to the partition, thus restoring you to your fresh install automatically in about 30 seconds, and without a single reboot. Of course, this procedure is great for regular backing up of NTFS partitions, too, and may be particularly useful before making major changes to your Windows installation, should things go wrong.

Moving clones

The method above works perfectly when you restore an image to where it originally came from. It also works fine if you want to move standard NTFS partitions about on a disk, or even to a new disk.

However, if you write a cloned image to a different place you will not be able to boot from the new partition. This is because Windows XP stores information in the boot sector of the partition about where the partition is on the disk. If you move the partition to anywhere else, this information no longer reflects the actual state, and rather than correct it, XP simply refuses to boot at all.

The ntfsclone documentation itself mentions this issue, stating, "ntfsclone is a filesystem, not a system utility. Its aim is only NTFS cloning, not Windows cloning".

The way around this is not for the faint-hearted as it involves editing raw data on the disk directly. It does work, however, and so long as you are careful about what you do, and follow the instructions fully, there should be no risk to your data.

The best way to get to grips with the rather complex procedure is to follow an example. With the command above, a 20GB NTFS partition (/dev/hda2) was cloned to a file named ntfs.img.

The first step in moving the data is to find out where on the hard drive the new partition physically sits. If you run the command `fdisk -ul /dev/hda`, you will see the partitions listed, together with the sector at which they start and end.

In this example we will assume that the cloned partition is going to be written to /dev/hda3, which is the same size as the original. The `fdisk` command shows that partition hda3 begins at sector 39294990. But our clone's boot sector still points to /dev/hda2.

If you refer back to the picture, you can see that the file command mentions 'hidden sectors 208845' for the cloned data, which matches the starting location of /dev/hda2 (as shown in the `fdisk` output). It is necessary to modify the data on the new partition so that it too matches the new position on the disk.

Unfortunately, things become more complicated at this point. We cannot simply write 39294990 onto the disk, it must first be converted to hexadecimal and the partition edited using a binary hex editor.

Convert the figure to hexadecimal using a utility such as `kcalc`, or just enter '39294990 to hex' in Google. For our example disk, the sector start in hexadecimal for /dev/hda2 is 0x32FCD and for /dev/hda3 is 0x257980E. After the cloned image is written to the new partition, run the `hexedit` tool on it to change the boot sector:

```
# hexedit /dev/hda3
```

Linux ntfsprogs

By Barry Shilliday

Note that you are now editing the contents of the disk directly, so take extra care. The bytes that must be changed are the four bytes beginning at position 0x1C, so move the cursor over to this point. These four bytes represent the starting sector of the partition.

To make things even more complicated, the hexadecimal pairs are stored in reverse order, so rather than 32FCD (the starting sector for hda2) we have CD 2F 03 00. In other words, the pattern AaBbCcDd is stored as DdCcBbAa, with zeroes to fill any space.

The hda3 partition begins at sector 0x257980E, as we saw above. This is therefore written 0E 98 57 02, and these figures need to replace the old values. Once edited, save and exit with Ctrl & X. The new partition is now bootable.

For this method to work, you must ensure that the drive letter in Windows doesn't change. If your original installation booted with drive C:, so must the new one. Windows is fussy about this. You must also change the partition type to NTFS. In our example it was marked as Linux; you can fix this with the fdisk program.

While this procedure is far from simple, it is very useful if your disk layout needs to change (such as adding a new drive) and you want to restore a previous installation. Note that these steps are necessary only when you want to move a bootable NTFS partition. If you don't need to boot the partition, you need only write the partition back with ntfsclone. The data will be fully available to Windows, whether the partition is bootable or not.