NAME

ntfs-3g – Third Generation Read/Write NTFS Driver

SYNOPSIS

ntfs-3g [-o *option***[,...]]** *volume mount_point*
mount –t ntfs-3g [-o *option***[,...]]** *volume mount_point*
lowntfs-3g [-o *option***[,...]]** *volume mount_point*
mount –t lowntfs-3g [-o *option***[,...]]** *volume mount_point*

DESCRIPTION

ntfs-3g is an NTFS driver, which can create, remove, rename, move files, directories, hard links, and streams; it can read and write files, including streams, sparse files and transparently compressed files; it can handle special files like symbolic links, devices, and FIFOs; moreover it provides standard management of file ownership and permissions, including POSIX ACLs.

It comes in two variants **ntfs-3g** and **lowntfs-3g** with a few differences mentioned below in relevant options descriptions.

The *volume* to be mounted can be either a block device or an image file.

Windows hibernation and fast restarting

On computers which can be dual-booted into Windows or Linux, Windows has to be fully shut down before booting into Linux, otherwise the NTFS file systems on internal disks may be left in an inconsistent state and changes made by Linux may be ignored by Windows.

So, Windows may not be left in hibernation when starting Linux, in order to avoid inconsistencies. Moreover, the fast restart feature available on recent Windows systems has to be disabled. This can be achieved by issuing as an Administrator the Windows command which disables both hibernation and fast restarting :

```
powercfg /h off
```

If either Windows is hibernated or its fast restart is enabled, partitions on internal disks are forced to be mounted in read-only mode.

Access Handling and Security

By default, files and directories are owned by the effective user and group of the mounting process, and everybody has full read, write, execution and directory browsing permissions. You can also assign permissions to a single user by using the **uid** and/or the **gid** options together with the **umask,** or **fmask** and **dmask** options.

Doing so, Windows users have full access to the files created by **ntfs-3g.**

But, by setting the **permissions** option, you can benefit from the full ownership and permissions features as defined by POSIX. Moreover, by defining a Windows-to-Linux user mapping, the ownerships and permissions are even applied to Windows users and conversely.

If **ntfs-3g** is set setuid-root then non-root users will be also able to mount volumes.

Windows Filename Compatibility

NTFS supports several filename namespaces: DOS, Win32 and POSIX. While the **ntfs-3g** driver handles all of them, it always creates new files in the POSIX namespace for maximum portability and interoperability reasons. This means that filenames are case sensitive and all characters are allowed except '/' and '\0'. This is perfectly legal on Windows, though some application may get confused. The option **windows_names** may be used to apply Windows restrictions to new file names.

Alternate Data Streams (ADS)

NTFS stores all data in streams. Every file has exactly one unnamed data stream and can have many named data streams. The size of a file is the size of its unnamed data stream. By default, **ntfs-3g** will only read the unnamed data stream.

By using the options "streams_interface=windows", with the ntfs-3g driver (not possible with lowntfs-3g), you will be able to read any named data streams, simply by specifying the stream's name after a colon. For

example:

cat some.mp3:artist

Named data streams act like normal files, so you can read from them, write to them and even delete them (using rm). You can list all the named data streams a file has by getting the "ntfs.streams.list" extended attribute.

OPTIONS

Below is a summary of the options that **ntfs-3g** accepts.

uid=*value* and **gid=***value*
Set the owner and the group of files and directories. The values are numerical. The defaults are the uid and gid of the current process.

umask=*value*
Set the bitmask of the file and directory permissions that are not present. The value is given in octal. The default value is 0 which means full access to everybody.

fmask=*value*
Set the bitmask of the file permissions that are not present. The value is given in octal. The default value is 0 which means full access to everybody.

dmask=*value*
Set the bitmask of the directory permissions that are not present. The value is given in octal. The default value is 0 which means full access to everybody.

usermapping= *file-name*
Use file *file-name* as the user mapping file instead of the default **.NTFS-3G/UserMapping**. If *file-name* defines a full path, the file must be located on a partition previously mounted. If it defines a relative path, it is interpreted relative to the root of NTFS partition being mounted.

When a user mapping file is defined, the options **uid=**, **gid=**, **umask=**, **fmask=**, **dmask=** and **silent** are ignored.

permissions
Set standard permissions on created files and use standard access control. This option is set by default when a user mapping file is present.

acl Enable setting Posix ACLs on created files and use them for access control. This option is only available on specific builds. It is set by default when a user mapping file is present and the **permissions** mount option is not set.

inherit When creating a new file, set its initial protections according to inheritance rules defined in parent directory. These rules deviate from Posix specifications, but yield a better Windows compatibility. The **permissions** option or a valid user mapping file is required for this option to be effective.

ro Mount filesystem read−only. Useful if Windows is hibernated or the NTFS journal file is unclean.

locale=*value*
This option can be useful when wanting a language specific locale environment. It is however discouraged as it leads to files with untranslatable chars to not be visible.

force This option is obsolete. It has been superseded by the **recover** and **norecover** options.

recover
Recover and try to mount a partition which was not unmounted properly by Windows. The Windows logfile is cleared, which may cause inconsistencies. Currently this is the default option.

norecover
Do not try to mount a partition which was not unmounted properly by Windows.

ignore_case (only with lowntfs-3g)
Ignore character case when accessing a file (**FOO**, **Foo**, **foo**, etc. designate the same file). All files are displayed with lower case in directory listings.

remove_hiberfile
When the NTFS volume is hibernated, a read-write mount is denied and a read-only mount is forced. One needs either to resume Windows and shutdown it properly, or use this option which will remove the Windows hibernation file. Please note, this means that the saved Windows session will be completely lost. Use this option under your own responsibility.

atime, noatime, relatime
The **atime** option updates inode access time for each access.

The **noatime** option disables inode access time updates which can speed up file operations and prevent sleeping (notebook) disks spinning up too often thus saving energy and disk lifetime.

The **relatime** option is very similar to **noatime.** It updates inode access times relative to modify or change time. The access time is only updated if the previous access time was earlier than the current modify or change time. Unlike **noatime** this option doesn't break applications that need to know if a file has been read since the last time it was modified. This is the default behaviour.

delay_mtime[= value]
Only update the file modification time and the file change time of a file when it is closed or when the indicated delay since the previous update has elapsed. The argument is a number of seconds, with a default value of 60. This is mainly useful for big files which are kept open for a long time and written to without changing their size, such as databases or file system images mounted as loop.

show_sys_files
Show the metafiles in directory listings. Otherwise the default behaviour is to hide the metafiles, which are special files used to store the NTFS structure. Please note that even when this option is specified, "$MFT" may not be visible due to a glibc bug. Furthermore, irrespectively of show_sys_files, all files are accessible by name, for example you can always do "ls –l '$UpCase'".

hide_hid_files
Hide the hidden files and directories in directory listings, the hidden files and directories being the ones whose NTFS attribute have the hidden flag set. The hidden files will not be selected when using wildcards in commands, but all files and directories remain accessible by full name, for example you can always display the Windows trash bin directory by : "ls –ld '$RECYCLE.BIN'".

hide_dot_files
Set the hidden flag in the NTFS attribute for created files and directories whose first character of the name is a dot. Such files and directories normally do not appear in directory listings, and when the flag is set they do not appear in Windows directory displays either. When a file is renamed or linked with a new name, the hidden flag is adjusted to the latest name.

windows_names
This option prevents files, directories and extended attributes to be created with a name not allowed by windows, because

- it contains some not allowed character,
- or the last character is a space or a dot,
- or the name is reserved.

The forbidden characters are the nine characters " * / : < > ? \ | and those whose code is less than 0x20, and the reserved names are CON, PRN, AUX, NUL, COM1..COM9, LPT1..LPT9, with no suffix or followed by a dot.

Existing such files can still be read (and renamed).

allow_other
This option overrides the security measure restricting file access to the user mounting the filesystem. This option is only allowed to root, but this restriction can be overridden by the 'user_allow_other' option in the /etc/fuse.conf file.

max_read=*value*
With this option the maximum size of read operations can be set. The default is infinite. Note that the size of read requests is limited anyway to 32 pages (which is 128kbyte on i386).

silent Do nothing, without returning any error, on chmod and chown operations and on permission checking errors, when the **permissions** option is not set and no user mapping file is defined. This option is on by default, and when set off (through option **no_def_opts**) ownership and permissions parameters have to be set.

no_def_opts
By default ntfs-3g acts as if "silent" (ignore permission errors when permissions are not enabled), "allow_other" (allow any user to access files) and "nonempty" (allow mounting on non-empty directories) were set, and "no_def_opts" cancels these default options.

streams_interface=*value*
This option controls how the user can access Alternate Data Streams (ADS) or in other words, named data streams. It can be set to, one of **none**, **windows** or **xattr**. If the option is set to **none**, the user will have no access to the named data streams. If it is set to **windows** (not possible with lowntfs-3g), then the user can access them just like in Windows (eg. cat file:stream). If it's set to **xattr**, then the named data streams are mapped to xattrs and user can manipulate them using **{get,set}fattr** utilities. The default is **xattr**.

user_xattr
Same as **streams_interface=***xattr*.

efs_raw
This option should only be used in backup or restore situation. It changes the apparent size of files and the behavior of read and write operation so that encrypted files can be saved and restored without being decrypted. The **user.ntfs.efsinfo** extended attribute has also to be saved and restored for the file to be decrypted.

compression
This option enables creating new transparently compressed files in directories marked for compression. A directory is marked for compression by setting the bit 11 (value 0x00000800) in its Windows attribute. In such a directory, new files are created compressed and new subdirectories are themselves marked for compression. The option and the flag have no effect on existing files. Currently this is the default option.

nocompression
This option disables creating new transparently compressed files in directories marked for compression. Existing compressed files can still be read and updated.

big_writes
This option prevents fuse from splitting write buffers into 4K chunks, enabling big write buffers to be transferred from the application in a single step (up to some system limit, generally 128K bytes).

debug Makes ntfs-3g to print a lot of debug output from libntfs-3g and FUSE.

no_detach
Makes ntfs-3g to not detach from terminal and print some debug output.

USER MAPPING

NTFS uses specific ids to record the ownership of files instead of the **uid** and **gid** used by Linux. As a consequence a mapping between the ids has to be defined for ownerships to be recorded into NTFS and recognized.

By default, this mapping is fetched from the file **.NTFS-3G/UserMapping** located in the NTFS partition. The option **usermapping=** may be used to define another location. When the option permissions is set and no mapping file is found, a default mapping is used.

Each line in the user mapping file defines a mapping. It is organized in three fields separated by colons. The first field identifies a **uid**, the second field identifies a **gid** and the third one identifies the corresponding NTFS id, known as a **SID**. The **uid** and the **gid** are optional and defining both of them for the same **SID** is not recommended.

If no interoperation with Windows is needed, you can use the option **permissions** to define a standard mapping. Alternately, you may define your own mapping by setting a single default mapping with no uid and gid. In both cases, files created on Linux will appear to Windows as owned by a foreign user, and files created on Windows will appear to Linux as owned by root. Just copy the example below and replace the 9 and 10-digit numbers by any number not greater than 4294967295. The resulting behavior is the same as the one with the option permission set with no ownership option and no user mapping file available.

::S-1-5-21-3141592653-589793238-462643383-10000

If a strong interoperation with Windows is needed, the mapping has to be defined for each user and group known in both system, and the **SID**s used by Windows has to be collected. This will lead to a user mapping file like :

john::S-1-5-21-3141592653-589793238-462643383-1008
mary::S-1-5-21-3141592653-589793238-462643383-1009
:smith:S-1-5-21-3141592653-589793238-462643383-513
::S-1-5-21-3141592653-589793238-462643383-10000

The utility **ntfsusermap** may be used to create such a user mapping file.

EXAMPLES

Mount /dev/sda1 to /mnt/windows:

ntfs-3g /dev/sda1 /mnt/windows

or

mount -t ntfs-3g /dev/sda1 /mnt/windows

Mount the ntfs data partition /dev/sda3 to /mnt/data with standard Linux permissions applied :

ntfs-3g -o permissions /dev/sda3 /mnt/data

or

mount -t ntfs-3g -o permissions /dev/sda3 /mnt/data

Read−only mount /dev/sda5 to /home/user/mnt and make user with uid 1000 to be the owner of all files:

ntfs-3g /dev/sda5 /home/user/mnt −o ro,uid=1000

/etc/fstab entry for the above (the sixth and last field has to be zero to avoid a file system check at boot time) :

/dev/sda5 /home/user/mnt ntfs−3g ro,uid=1000 0 0

Unmount /mnt/windows:

umount /mnt/windows

EXIT CODES

To facilitate the use of the **ntfs-3g** driver in scripts, an exit code is returned to give an indication of the mountability status of a volume. Value 0 means success, and all other ones mean an error. The unique error codes are documented in the **ntfs-3g.probe**(8) manual page.

KNOWN ISSUES

Please see

http://www.tuxera.com/support/

for common questions and known issues. If you would find a new one in the latest release of the software then please send an email describing it in detail. You can contact the development team on the ntfs−3g−devel@lists.sf.net address.

AUTHORS

ntfs-3g was based on and a major improvement to ntfsmount and libntfs which were written by Yura Pakhuchiy and the Linux-NTFS team. The improvements were made, the ntfs-3g project was initiated and currently led by long time Linux-NTFS team developer Szabolcs Szakacsits (szaka@tuxera.com).

THANKS

Several people made heroic efforts, often over five or more years which resulted the ntfs-3g driver. Most importantly they are Anton Altaparmakov, Jean-Pierre André, Richard Russon, Szabolcs Szakacsits, Yura Pakhuchiy, Yuval Fledel, and the author of the groundbreaking FUSE filesystem development framework, Miklos Szeredi.

SEE ALSO

ntfs-3g.probe(8), **ntfsprogs**(8), **attr**(5), **getfattr**(1)