NAME

ntfsresize - resize an NTFS filesystem without data loss

SYNOPSIS

ntfsresize [*OPTIONS*] **--info(-mb-only)** *DEVICE*
ntfsresize [*OPTIONS*] [**--size** *SIZE*[**k**|**M**|**G**]] *DEVICE*

DESCRIPTION

The **ntfsresize** program safely resizes Windows XP, Windows Server 2003, Windows 2000, Windows NT4 and Longhorn NTFS filesystems without data loss. All NTFS versions are supported, used by 32-bit and 64-bit Windows. **Defragmentation is NOT required prior to resizing** because the program can relocate any data if needed, without risking data integrity.

Ntfsresize can be used to shrink or enlarge any NTFS filesystem located on an unmounted *DEVICE* (usually a disk partition). The new filesystem will fit in a DEVICE whose desired size is *SIZE* bytes. The *SIZE* parameter may have one of the optional modifiers **k**, **M**, **G**, which means the *SIZE* parameter is given in kilo-, mega- or gigabytes respectively. **Ntfsresize** conforms to the SI, ATA, IEEE standards and the disk manufacturers by using k=10^3, M=10^6 and G=10^9.

If both **--info(-mb-only)** and **--size** are omitted then the NTFS filesystem will be enlarged to match the underlying *DEVICE* size.

To resize a filesystem on a partition, you must resize BOTH the filesystem and the partition by editing the partition table on the disk. Similarly to other command line filesystem resizers, **ntfsresize** doesn't manipulate the size of the partitions, hence to do that you must use a disk partitioning tool as well, for example fdisk(8). Alternatively you could use one of the many user friendly partitioners that uses **ntfsresize** internally, like Mandriva's DiskDrake, QTParted, SUSE/Novell's YaST Partitioner, IBM's EVMS, GParted or Debian/Ubuntu's Partman.

IMPORTANT! It's a good practice making REGULAR BACKUPS of your valuable data, especially before using ANY partitioning tools. To do so for NTFS, you could use ntfsclone(8). Don't forget to save the partition table as well!

Shrinkage

If you wish to shrink an NTFS partition, first use **ntfsresize** to shrink the size of the filesystem. Then you could use fdisk(8) to shrink the size of the partition by deleting the partition and recreating it with the smaller size. Do not make the partition smaller than the new size of NTFS otherwise you won't be able to boot. If you did so notwithstanding then just recreate the partition to be as large as NTFS.

Enlargement

To enlarge an NTFS filesystem, first you must enlarge the size of the underlying partition. This can be done using fdisk(8) by deleting the partition and recreating it with a larger size. Make sure it will not overlap with another existing partition. You may enlarge upwards (first sector unchanged) or downwards (last sector unchanged), but you may not enlarge at both ends in a single step. If you merge two NTFS partitions, only one of them can be expanded to the merged partition. After you have enlarged the partition, you may use **ntfsresize** to enlarge the size of the filesystem.

Partitioning

When recreating the partition by a disk partitioning tool, make sure you create it at the same starting sector and with the same partition type as before. Otherwise you won't be able to access your filesystem. Use the 'u' fdisk command to switch to the reliable sector unit from the default cylinder one.

Also make sure you set the bootable flag for the partition if it existed before. Failing to do so you might not be able to boot your computer from the disk.

OPTIONS

Below is a summary of all the options that **ntfsresize** accepts. Nearly all options have two equivalent names. The short name is preceded by **-** and the long name is preceded by **--**. Any single letter options, that don't take an argument, can be combined into a single command, e.g. **-fv** is equi valent to **-f -v**. Long named options can be abbreviated to any unique prefix of their name.

-c, **--check**

By using this option ntfsresize will only check the device to ensure that it is ready to be resized. If not, it will print any errors detected. If the device is fine, nothing will be printed.

-i, **--info**

By using this option without **--expand**, ntfsresize will determine the theoretically smallest shrunken filesystem size supported. Most of the time the result is the space already used on the filesystem. Ntfsresize will refuse shrinking to a smaller size than what you got by this option and depending on several factors it might be unable to shrink very close to this theoretical size. Although the integrity of your data should be never in risk, it's still strongly recommended to make a test run by using the **--no-action** option before real resizing.

Practically the smallest shrunken size generally is at around "used space" + (20-200 MB). Please also take into account that Windows might need about 50-100 MB free space left to boot safely.

If used in association with option **--expand**, ntfsresize will determine the smallest downwards expansion size and the possible increments to the size. These are exact byte counts which must not be rounded. This option may be used after the partition has been expanded provided the upper bound has not been changed.

This option never causes any changes to the filesystem, the partition is opened read-only.

-m, **--info-mb-only**

Like the info option, only print out the shrinkable size in MB. Print nothing if the shrink size is the same as the original size (in MB). This option cannot be used in association with option **--expand**.

-s, **--size** SIZE[**k**|**M**|**G**]

Resize filesystem to fit in a partition whose size is *SIZE*[**k**|**M**|**G**] bytes by shifting its end and keeping its beginning unchanged. The filesystem size is set to be at least one sector smaller than the partition. The optional modifiers**k**, **M**, **G** mean the *SIZE* parameter is gi ven in kilo-, mega- or gigabytes respectively. Conforming to standards, k=10^3, M=10^6 and G=10^9. ki=2^10, Mi=2^20 and Gi=2^30 are also allowed. Use this option with **--no-action** first.

-x, **--expand**

Expand the filesystem to the current partition size, shifting down its beginning and keeping its end unchanged. The metadata is recreated in the expanded space and no user data is relocated. This is incompatible with option -s (or --size) and can only be made if the expanded space is an exact multiple of the cluster size. It must also be large enough to hold the new metadata.

If the expansion is interrupted for some reason (power outage, etc), you may restart the resizing, as the original data and metadata have been kept unchanged.

Note : expanding a Windows system partition and filesystem downwards may lead to the registry or some files not matching the new system layout, or to some important files being located too far from the beginning of the partition, thus making Windows not bootable.

-f, **--force**

Forces ntfsresize to proceed with the resize operation either without prompting for an explicit acceptance, or if the filesystem is marked for consistency check. Double the option (-ff, --force --force) to avoid prompting even if the file system is marked for check.

Please note, ntfsresize always marks the filesystem for consistency check before a real resize operation and it leaves that way for extra safety. Thus if NTFS was marked by ntfsresize then it's safe to use this option. If you need to resize several times without booting into Windows between each resizing steps then you must use this option.

-n, --no-action

Use this option to make a test run before doing the real resize operation. Volume will be opened read-only and **ntfsresize** displays what it would do if it were to resize the filesystem. Continue with the real resizing only if the test run passed.

-b, **--bad-sectors**
Support disks having hardware errors, bad sectors with those **ntfsresize** would refuse to work by default.

Prior using this option, it's strongly recommended to make a backup by ntfsclone(8) using the --rescue option, then running 'chkdsk /f /r volume:' on Windows from the command line. If the disk guarantee is still valid then replace it. It's defected. Please also note, that no software can repair these type of hardware errors. The most what they can do is to work around the permanent defects.

This option doesn't have any effect if the disk is flawless.

-P, **--no-progress-bar**
Don't show progress bars.

-v, **--verbose**
More output.

-V, **--version**
Print the version number of **ntfsresize** and exit.

-h, **--help**
Display help and exit.

EXIT CODES

The exit code is 0 on success, non-zero otherwise.

KNOWN ISSUES

No reliability problem is known. If you need help please try the Ntfsresize FAQ first (see below) and if you don't find your answer then send your question, comment or bug report to the development team:
ntfs-3g-devel@lists.sf.net

There are a few very rarely met restrictions at present: filesystems having unknown bad sectors, relocation of the first MFT extent and resizing into the middle of a $MFTMirr extent aren't supported yet. These cases are detected and resizing is restricted to a safe size or the closest safe size is displayed.

Ntfsresize schedules an NTFS consistency check and after the first boot into Windows you must see **chkdsk** running on a blue background. This is intentional and no need to worry about it. Windows may force a quick reboot after the consistency check. Moreover after repartitioning your disk and depending on the hardware configuration, the Windows message **System Settings Change** may also appear. Just acknowledge it and reboot again.

The disk geometry handling semantic (HDIO_GETGEO ioctl) has changed in an incompatible way in Linux 2.6 kernels and this triggered multitudinous partition table corruptions resulting in unbootable Windows systems, even if NTFS was consistent, if parted(8) was involved in some way. This problem was often attributed to ntfsresize but in fact it's completely independent of NTFS thus ntfsresize. Moreover ntfsresize never touches the partition table at all. By changing the 'Disk Access Mode' to LBA in the BIOS makes booting work again, most of the time. You can find more information about this issue in the Troubleshooting section of the below referred Ntfsresize FAQ.

AUTHORS

ntfsresize was written by Szabolcs Szakacsits, with contributions from Anton Altaparmakov and Richard Russon. It was ported to ntfs-3g by Erik Larsson and Jean-Pierre Andre.

ACKNOWLEDGEMENT

Many thanks to Anton Altaparmakov and Richard Russon for libntfs, the excellent documentation and comments, to Gergely Madarasz, Dewey M. Sasser and Miguel Lastra and his colleagues at the University of Granada for their continuous and highly valuable help, furthermore to Erik Meade, Martin Fick, Sandro Hawke, Dave Croal, Lorrin Nelson, Geert Hendrickx, Robert Bjorkman and Richard Burdick for beta testing the relocation support, to Florian Eyben, Fritz Oppliger, Richard Ebling, Sid-Ahmed Touati, Jan Kiszka, Benjamin Redelings, Christopher Haney, Ryan Durk, Ralf Beyer, Scott Hansen, Alan Evans for the valued contributions and to Theodore Ts'o whose resize2fs(8) man page originally formed the basis of this

page.

AVAILABILITY

ntfsresize is part of the **ntfs-3g** package and is available from:
http://www.tuxera.com/community/

Ntfsresize related news, example of usage, troubleshooting, statically linked binary and FAQ (frequently asked questions) are maintained at:
http://mlf.linux.rulez.org/mlf/ezaz/ntfsresize.html

SEE ALSO

fdisk(8), cfdisk(8), sfdisk(8), parted(8), **evms(8),** ntfsclone(8), mkntfs(8), ntfsprogs(8)