

**NAME**

resize2fs – ext2/ext3/ext4 file system resizer

**SYNOPSIS**

**resize2fs** [ **-fFpPMbs** ] [ **-d** *debug-flags* ] [ **-S** *RAID-stride* ] [ **-z** *undo\_file* ] *device* [ *size* ]

**DESCRIPTION**

The **resize2fs** program will resize ext2, ext3, or ext4 file systems. It can be used to enlarge or shrink an unmounted file system located on *device*. If the file system is mounted, it can be used to expand the size of the mounted file system, assuming the kernel and the file system supports on-line resizing. (Modern Linux 2.6 kernels will support on-line resize for file systems mounted using ext3 and ext4; ext3 file systems will require the use of file systems with the `resize_inode` feature enabled.)

The *size* parameter specifies the requested new size of the file system. If no units are specified, the units of the *size* parameter shall be the file system blocksize of the file system. Optionally, the *size* parameter may be suffixed by one of the following units designators: 'K', 'M', 'G', 'T' (either upper-case or lower-case) or 's' for power-of-two kilobytes, megabytes, gigabytes, terabytes or 512 byte sectors respectively. The *size* of the file system may never be larger than the size of the partition. If *size* parameter is not specified, it will default to the size of the partition.

The **resize2fs** program does not manipulate the size of partitions. If you wish to enlarge a file system, you must make sure you can expand the size of the underlying partition first. This can be done using **fdisk**(8) by deleting the partition and recreating it with a larger size or using **lvextend**(8), if you're using the logical volume manager **lvm**(8). When recreating the partition, make sure you create it with the same starting disk cylinder as before! Otherwise, the resize operation will certainly not work, and you may lose your entire file system. After running **fdisk**(8), run **resize2fs** to resize the ext2 file system to use all of the space in the newly enlarged partition.

If you wish to shrink an ext2 partition, first use **resize2fs** to shrink the size of file system. Then you may use **fdisk**(8) to shrink the size of the partition. When shrinking the size of the partition, make sure you do not make it smaller than the new size of the ext2 file system!

The **-b** and **-s** options enable and disable the 64bit feature, respectively. The **resize2fs** program will, of course, take care of resizing the block group descriptors and moving other data blocks out of the way, as needed. It is not possible to resize the file system concurrent with changing the 64bit status.

**OPTIONS**

- b** Turns on the 64bit feature, resizes the group descriptors as necessary, and moves other metadata out of the way.
- d** *debug-flags* Turns on various **resize2fs** debugging features, if they have been compiled into the binary. *debug-flags* should be computed by adding the numbers of the desired features from the following list:
 

2	– Debug block relocations
4	– Debug inode relocations
8	– Debug moving the inode table
16	– Print timing information
32	– Debug minimum file system size ( <b>-M</b> ) calculation
- f** Forces **resize2fs** to proceed with the file system resize operation, overriding some safety checks which **resize2fs** normally enforces.
- F** Flush the file system device's buffer caches before beginning. Only really useful for doing **resize2fs** time trials.
- M** Shrink the file system to minimize its size as much as possible, given the files stored in the file system.
- p** Print out percentage completion bars for each **resize2fs** phase during an offline (non-trivial) resize operation, so that the user can keep track of what the program is doing. (For very fast resize operations, no progress bars may be displayed.)

- P** Print an estimate of the number of file system blocks in the file system if it is shrunk using **resize2fs**'s **-M** option and then exit.
- s** Turns off the 64bit feature and frees blocks that are no longer in use.
- S RAID-stride**  
The **resize2fs** program will heuristically determine the RAID stride that was specified when the file system was created. This option allows the user to explicitly specify a RAID stride setting to be used by **resize2fs** instead.
- z undo\_file**  
Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the **undo\_file** argument, the undo file will be written to a file named **resize2fs-device.e2undo** in the directory specified via the **E2FSPROGS\_UNDO\_DIR** environment variable.

WARNING: The undo file cannot be used to recover from a power or system crash.

## KNOWN BUGS

The minimum size of the file system as estimated by **resize2fs** may be incorrect, especially for file systems with 1k and 2k block sizes.

## AUTHOR

**resize2fs** was written by Theodore Ts'o <tytso@mit.edu>.

## COPYRIGHT

**Resize2fs** is Copyright 1998 by Theodore Ts'o and PowerQuest, Inc. All rights reserved. As of April, 2000 **Resize2fs** may be redistributed under the terms of the GPL.

## SEE ALSO

**fdisk(8)**, **e2fsck(8)**, **mke2fs(8)**, **lvm(8)**, **lvextend(8)**