



## **Rocky Enterprise Linux 9.2 Manual Pages on command 'mkfs.ext3.8'**

**C:\>man mkfs.ext3.8**

MKE2FS(8)                      System Manager's Manual                      MKE2FS(8)

### NAME

mkfs2fs - create an ext2/ext3/ext4 filesystem

### SYNOPSIS

```
mkfs2fs [ -c | -l filename ] [ -b block-size ] [ -C cluster-size ] [ -d root-direct?
tory ] [ -D ] [ -g blocks-per-group ] [ -G number-of-groups ] [ -i bytes-per-inode
] [ -l inode-size ] [ -j ] [ -J journal-options ] [ -N number-of-inodes ] [ -n ] [
-m reserved-blocks-percentage ] [ -o creator-os ] [ -O [^]feature[,...] ] [ -q ] [
-r fs-revision-level ] [ -E extended-options ] [ -v ] [ -F ] [ -L volume-label ] [
-M last-mounted-directory ] [ -S ] [ -t fs-type ] [ -T usage-type ] [ -U UUID ] [
-V ] [ -e errors-behavior ] [ -z undo_file ] device [ fs-size ]

mkfs2fs -O journal_dev [ -b block-size ] [ -L volume-label ] [ -n ] [ -q ] [ -v ]
external-journal [ fs-size ]
```

### DESCRIPTION

mkfs2fs is used to create an ext2, ext3, or ext4 filesystem, usually in a disk partition (or file) named by device.

The file system size is specified by fs-size. If fs-size does not have a suffix, it is interpreted as power-of-two kilobytes, unless the -b blocksize option is specified, in which case fs-size is interpreted as the number of blocksize blocks. If the fs-size is suffixed by 'k', 'm', 'g', 't' (either upper-case or lower-case), then it is interpreted in power-of-two kilobytes, megabytes, gigabytes, terabytes, etc. If fs-size is omitted, mkfs2fs will create the file system based on the device

size.

If `mke2fs` is run as `mkfs.XXX` (i.e., `mkfs.ext2`, `mkfs.ext3`, or `mkfs.ext4`) the option `-t XXX` is implied; so `mkfs.ext3` will create a file system for use with `ext3`, `mkfs.ext4` will create a file system for use with `ext4`, and so on.

The defaults of the parameters for the newly created filesystem, if not overridden by the options listed below, are controlled by the `/etc/mke2fs.conf` configuration file. See the `mke2fs.conf(5)` manual page for more details.

## OPTIONS

### `-b` block-size

Specify the size of blocks in bytes. Valid block-size values are 1024, 2048 and 4096 bytes per block. If omitted, block-size is heuristically determined by the filesystem size and the expected usage of the filesystem (see the `-T` option). If block-size is preceded by a negative sign (`'-'`), then `mke2fs` will use heuristics to determine the appropriate block size, with the constraint that the block size will be at least block-size bytes. This is useful for certain hardware devices which require that the blocksize be a multiple of 2k.

`-c` Check the device for bad blocks before creating the file system. If this option is specified twice, then a slower read-write test is used instead of a fast read-only test.

### `-C` cluster-size

Specify the size of cluster in bytes for filesystems using the bigalloc feature. Valid cluster-size values are from 2048 to 256M bytes per cluster. This can only be specified if the bigalloc feature is enabled. (See the `ext4 (5)` man page for more details about bigalloc.) The default cluster size if bigalloc is enabled is 16 times the block size.

### `-d` root-directory

Copy the contents of the given directory into the root directory of the filesystem.

`-D` Use direct I/O when writing to the disk. This avoids `mke2fs` dirtying a lot of buffer cache memory, which may impact other applications running on a busy server. This option will cause `mke2fs` to run much more slowly, however, so there is a tradeoff to using direct I/O.

## -e error-behavior

Change the behavior of the kernel code when errors are detected. In all cases, a filesystem error will cause `e2fsck(8)` to check the filesystem on the next boot. `error-behavior` can be one of the following:

`continue` Continue normal execution.

`remount-ro` Remount filesystem read-only.

`panic` Cause a kernel panic.

## -E extended-options

Set extended options for the filesystem. Extended options are comma separated, and may take an argument using the equals (=) sign. The `-E` option used to be `-R` in earlier versions of `mke2fs`. The `-R` option is still accepted for backwards compatibility, but is deprecated. The following extended options are supported:

`encoding=encoding-name`

Enable the casefold feature in the super block and set `encoding-name` as the encoding to be used. If `encoding-name` is not specified, the encoding defined in `mke2fs.conf(5)` is used.

`encoding_flags=encoding-flags`

Define parameters for file name character encoding operations. If a flag is not changed using this parameter, its default value is used. `encoding-flags` should be a comma-separated lists of flags to be enabled. To disable a flag, add it to the list with the prefix "no".

The only flag that can be set right now is `strict` which means that invalid strings should be rejected by the file system. In the default configuration, the `strict` flag is disabled.

`mmp_update_interval=interval`

Adjust the initial MMP update interval to `interval` seconds.

Specifying an interval of 0 means to use the default interval.

The specified interval must be less than 300 seconds. Requires that the `mmp` feature be enabled.

`stride=stride-size`

Configure the filesystem for a RAID array with `stride-size`

filesystem blocks. This is the number of blocks read or written to disk before moving to the next disk, which is sometimes referred to as the chunk size. This mostly affects placement of filesystem metadata like bitmaps at mke2fs time to avoid placing them on a single disk, which can hurt performance. It may also be used by the block allocator.

`stripe_width=stripe-width`

Configure the filesystem for a RAID array with stripe-width filesystem blocks per stripe. This is typically  $\text{stride-size} * N$ , where  $N$  is the number of data-bearing disks in the RAID (e.g. for RAID 5 there is one parity disk, so  $N$  will be the number of disks in the array minus 1). This allows the block allocator to prevent read-modify-write of the parity in a RAID stripe if possible when the data is written.

`offset=offset`

Create the filesystem at an offset from the beginning of the device or file. This can be useful when creating disk images for virtual machines.

`resize=max-online-resize`

Reserve enough space so that the block group descriptor table can grow to support a filesystem that has max-online-resize blocks.

`lazy_itable_init[= <0 to disable, 1 to enable>]`

If enabled and the `uninit_bg` feature is enabled, the inode table will not be fully initialized by mke2fs. This speeds up filesystem initialization noticeably, but it requires the kernel to finish initializing the filesystem in the background when the filesystem is first mounted. If the option value is omitted, it defaults to 1 to enable lazy inode table zeroing.

`lazy_journal_init[= <0 to disable, 1 to enable>]`

If enabled, the journal inode will not be fully zeroed out by mke2fs. This speeds up filesystem initialization noticeably, but carries some small risk if the system crashes before the

journal has been overwritten entirely one time. If the option value is omitted, it defaults to 1 to enable lazy journal inode zeroing.

#### no\_copy\_xattrs

Normally `mke2fs` will copy the extended attributes of the files in the directory hierarchy specified via the (optional) `-d op?` option. This will disable the copy and leaves the files in the newly created file system without any extended attributes.

#### num\_backup\_sb=<0|1|2>

If the `sparse_super2` file system feature is enabled this option controls whether there will be 0, 1, or 2 backup superblocks created in the file system.

#### packed\_meta\_blocks[= <0 to disable, 1 to enable>]

Place the allocation bitmaps and the inode table at the beginning of the disk. This option requires that the `flex_bg` file system feature to be enabled in order for it to have effect, and will also create the journal at the beginning of the file system. This option is useful for flash devices that use SLC flash at the beginning of the disk. It also maximizes the range of contiguous data blocks, which can be useful for certain specialized use cases, such as supported Shingled Drives.

#### root\_owner[=uid:gid]

Specify the numeric user and group ID of the root directory. If no UID:GID is specified, use the user and group ID of the user running `mke2fs`. In `mke2fs` 1.42 and earlier the UID and GID of the root directory were set by default to the UID and GID of the user running the `mke2fs` command. The `root_owner=` option allows explicitly specifying these values, and avoid side-effects for users that do not expect the contents of the filesystem to change based on the user running `mke2fs`.

#### test\_fs

Set a flag in the filesystem superblock indicating that it may be mounted using experimental kernel code, such as the `ext4dev`

filesystem.

#### discard

Attempt to discard blocks at mkfs time (discarding blocks initially is useful on solid state devices and sparse / thin-provisioned storage). When the device advertises that discard also zeroes data (any subsequent read after the discard and before write returns zero), then mark all not-yet-zeroed inode tables as zeroed. This significantly speeds up filesystem initialization. This is set as default.

#### nodiscard

Do not attempt to discard blocks at mkfs time.

#### quotatype

Specify the which quota types (usrquota, grpquota, prjquota) which should be enabled in the created file system. The argument of this extended option should be a colon separated list. This option has effect only if the quota feature is set. The default quota types to be initialized if this option is not specified is both user and group quotas. If the project feature is enabled that project quotas will be initialized as well.

-F Force mke2fs to create a filesystem, even if the specified device is not a partition on a block special device, or if other parameters do not make sense. In order to force mke2fs to create a filesystem even if the filesystem appears to be in use or is mounted (a truly dangerous thing to do), this option must be specified twice.

#### -g blocks-per-group

Specify the number of blocks in a block group. There is generally no reason for the user to ever set this parameter, as the default is optimal for the filesystem. (For administrators who are creating filesystems on RAID arrays, it is preferable to use the stride RAID parameter as part of the -E option rather than manipulating the number of blocks per group.) This option is generally used by developers who are developing test cases. If the bigalloc feature is enabled, the -g option will specify the number of clusters in a block group.

#### -G number-of-groups

Specify the number of block groups that will be packed together to create a larger virtual block group (or "flex\_bg group") in an ext4 filesystem. This improves meta-data locality and performance on meta-data heavy workloads. The number of groups must be a power of 2 and may only be specified if the flex\_bg filesystem feature is enabled.

#### -i bytes-per-inode

Specify the bytes/inode ratio. mke2fs creates an inode for every bytes-per-inode bytes of space on the disk. The larger the bytes-per-inode ratio, the fewer inodes will be created. This value generally shouldn't be smaller than the blocksize of the filesystem, since in that case more inodes would be made than can ever be used. Be warned that it is not possible to change this ratio on a filesystem after it is created, so be careful deciding the correct value for this parameter. Note that resizing a filesystem changes the number of inodes to maintain this ratio.

#### -l inode-size

Specify the size of each inode in bytes. The inode-size value must be a power of 2 larger or equal to 128. The larger the inode-size the more space the inode table will consume, and this reduces the usable space in the filesystem and can also negatively impact performance. It is not possible to change this value after the filesystem is created.

In kernels after 2.6.10 and some earlier vendor kernels it is possible to utilize inodes larger than 128 bytes to store extended attributes for improved performance. Extended attributes stored in large inodes are not visible with older kernels, and such filesystems will not be mountable with 2.4 kernels at all.

The default inode size is controlled by the mke2fs.conf(5) file. In the mke2fs.conf file shipped with e2fsprogs, the default inode size is 256 bytes for most file systems, except for small file systems where the inode size will be 128 bytes.

#### -j Create the filesystem with an ext3 journal. If the -J option is not specified, the default journal parameters will be used to create an appropriately sized journal (given the size of the filesystem) stored within the filesystem?

tem. Note that you must be using a kernel which has ext3 support in order to actually make use of the journal.

#### -J journal-options

Create the ext3 journal using options specified on the command-line. Journal options are comma separated, and may take an argument using the equals (=) sign. The following journal options are supported:

`size=journal-size`

Create an internal journal (i.e., stored inside the filesystem) of size journal-size megabytes. The size of the journal must be at least 1024 filesystem blocks (i.e., 1MB if using 1k blocks, 4MB if using 4k blocks, etc.) and may be no more than 10,240,000 filesystem blocks or half the total file system size (whichever is smaller)

`location=journal-location`

Specify the location of the journal. The argument journal-location can either be specified as a block number, or if the number has a units suffix (e.g., 'M', 'G', etc.) interpret it as the offset from the beginning of the file system.

`device=external-journal`

Attach the filesystem to the journal block device located on external-journal. The external journal must already have been created using the command

```
mke2fs -O journal_dev external-journal
```

Note that external-journal must have been created with the same block size as the new filesystem. In addition, while there is support for attaching multiple filesystems to a single external journal, the Linux kernel and e2fsck(8) do not currently support shared external journals yet.

Instead of specifying a device name directly, external-journal can also be specified by either LABEL=label or UUID=UUID to locate the external journal by either the volume label or UUID stored in the ext2 superblock at the start of the journal. Use dumpe2fs(8) to display a journal device's volume label and UUID.

See also the `-L` option of `tune2fs(8)`.

Only one of the size or device options can be given for a filesystem.

`-l filename`

Read the bad blocks list from `filename`. Note that the block numbers in the bad block list must be generated using the same block size as used by `mke2fs`. As a result, the `-c` option to `mke2fs` is a much simpler and less error-prone method of checking a disk for bad blocks before formatting it, as `mke2fs` will automatically pass the correct parameters to the `badblocks` program.

`-L new-volume-label`

Set the volume label for the filesystem to `new-volume-label`. The maximum length of the volume label is 16 bytes.

`-m reserved-blocks-percentage`

Specify the percentage of the filesystem blocks reserved for the super-user. This avoids fragmentation, and allows root-owned daemons, such as `syslogd(8)`, to continue to function correctly after non-privileged processes are prevented from writing to the filesystem. The default percentage is 5%.

`-M last-mounted-directory`

Set the last mounted directory for the filesystem. This might be useful for the sake of utilities that key off of the last mounted directory to determine where the filesystem should be mounted.

`-n` Causes `mke2fs` to not actually create a filesystem, but display what it would do if it were to create a filesystem. This can be used to determine the location of the backup superblocks for a particular filesystem, so long as the `mke2fs` parameters that were passed when the filesystem was originally created are used again. (With the `-n` option added, of course!)

`-N number-of-inodes`

Overrides the default calculation of the number of inodes that should be reserved for the filesystem (which is based on the number of blocks and the bytes-per-inode ratio). This allows the user to specify the number of desired inodes directly.

`-o creator-os`

Overrides the default value of the "creator operating system" field of the

filesystem. The creator field is set by default to the name of the OS the mke2fs executable was compiled for.

**-O** [^]feature[,...]

Create a filesystem with the given features (filesystem options), overriding the default filesystem options. The features that are enabled by default are specified by the `base_features` relation, either in the `[defaults]` section in the `/etc/mke2fs.conf` configuration file, or in the `[fs_types]` subsections for the usage types as specified by the `-T` option, further modified by the `features` relation found in the `[fs_types]` subsections for the filesystem and usage types. See the `mke2fs.conf(5)` manual page for more details. The filesystem type-specific configuration setting found in the `[fs_types]` section will override the global default found in `[defaults]`. The filesystem feature set will be further edited using either the feature set specified by this option, or if this option is not given, by the `default_features` relation for the filesystem type being created, or in the `[defaults]` section of the configuration file.

The filesystem feature set is comprised of a list of features, separated by commas, that are to be enabled. To disable a feature, simply prefix the feature name with a caret (^) character. Features with dependencies will not be removed successfully. The pseudo-filesystem feature "none" will clear all filesystem features.

For more information about the features which can be set, please see the manual page `ext4(5)`.

**-q** Quiet execution. Useful if `mke2fs` is run in a script.

**-r** revision

Set the filesystem revision for the new filesystem. Note that 1.2 kernels only support revision 0 filesystems. The default is to create revision 1 filesystems.

**-S** Write superblock and group descriptors only. This is an extreme measure to be taken only in the very unlikely case that all of the superblock and backup superblocks are corrupted, and a last-ditch recovery method is desired by experienced users. It causes `mke2fs` to reinitialize the superblock and group descriptors, while not touching the inode table and the block and

inode bitmaps. The e2fsck program should be run immediately after this operation is used, and there is no guarantee that any data will be salvageable. Due to the wide variety of possible options to mke2fs that affect the on-disk layout, it is critical to specify exactly the same format options, such as blocksize, fs-type, feature flags, and other tunables when using this operation, or the filesystem will be further corrupted. In some cases, such as filesystems that have been resized, or have had features enabled after format time, it is impossible to overwrite all of the superblocks correctly, and at least some filesystem corruption will occur. It is best to run this on a full copy of the filesystem so other options can be tried if this doesn't work.

#### **-t fs-type**

Specify the filesystem type (i.e., ext2, ext3, ext4, etc.) that is to be created. If this option is not specified, mke2fs will pick a default either via how the command was run (for example, using a name of the form mkfs.ext2, mkfs.ext3, etc.) or via a default as defined by the `/etc/mke2fs.conf` file. This option controls which filesystem options are used by default, based on the `fstypes` configuration stanza in `/etc/mke2fs.conf`.

If the `-O` option is used to explicitly add or remove filesystem options that should be set in the newly created filesystem, the resulting filesystem may not be supported by the requested fs-type. (e.g., "mke2fs -t ext3 -O extent /dev/sdXX" will create a filesystem that is not supported by the ext3 implementation as found in the Linux kernel; and "mke2fs -t ext3 -O ^has\_journal /dev/hdXX" will create a filesystem that does not have a journal and hence will not be supported by the ext3 filesystem code in the Linux kernel.)

#### **-T usage-type[,...]**

Specify how the filesystem is going to be used, so that mke2fs can choose optimal filesystem parameters for that use. The usage types that are supported are defined in the configuration file `/etc/mke2fs.conf`. The user may specify one or more usage types using a comma separated list.

If this option is not specified, mke2fs will pick a single default usage type based on the size of the filesystem to be created. If the filesystem

size is less than 3 megabytes, mke2fs will use the filesystem type floppy. If the filesystem size is greater than or equal to 3 but less than 512 megabytes, mke2fs(8) will use the filesystem type small. If the filesystem size is greater than or equal to 4 terabytes but less than 16 terabytes, mke2fs(8) will use the filesystem type big. If the filesystem size is greater than or equal to 16 terabytes, mke2fs(8) will use the filesystem type huge. Otherwise, mke2fs(8) will use the default filesystem type default.

#### -U UUID

Set the universally unique identifier (UUID) of the filesystem to UUID. The format of the UUID is a series of hex digits separated by hyphens, like this: "c1b9d5a2-f162-11cf-9ece-0020afc76f16". The UUID parameter may also be one of the following:

clear clear the filesystem UUID

random generate a new randomly-generated UUID

time generate a new time-based UUID

-v Verbose execution.

-V Print the version number of mke2fs and exit.

#### -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 mke2fs-device.e2undo in the directory specified via the E2FSPROGS\_UNDO\_DIR environment variable or the undo\_dir directive in the configuration file.

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

## ENVIRONMENT

### MKE2FS\_SYNC

If set to non-zero integer value, its value is used to determine how often sync(2) is called during inode table initialization.

### MKE2FS\_CONFIG

Determines the location of the configuration file (see `mke2fs.conf(5)`).

#### MKE2FS\_FIRST\_META\_BG

If set to non-zero integer value, its value is used to determine first meta block group. This is mostly for debugging purposes.

#### MKE2FS\_DEVICE\_SECTSIZE

If set to non-zero integer value, its value is used to determine logical sector size of the device.

#### MKE2FS\_DEVICE\_PHYS\_SECTSIZE

If set to non-zero integer value, its value is used to determine physical sector size of the device.

#### MKE2FS\_SKIP\_CHECK\_MSG

If set, do not show the message of filesystem automatic check caused by mount count or check interval.

#### AUTHOR

This version of `mke2fs` has been written by Theodore Ts'o <[tytso@mit.edu](mailto:tytso@mit.edu)>.

#### AVAILABILITY

`mke2fs` is part of the `e2fsprogs` package and is available from <http://e2fsprogs.sourceforge.net>.

#### SEE ALSO

`mke2fs.conf(5)`, `badblocks(8)`, `dumpe2fs(8)`, `e2fsck(8)`, `tune2fs(8)`, `ext4(5)`

E2fsprogs version 1.45.5

January 2020

MKE2FS(8)