NAME

mke2fs.conf − Configuration file for mke2fs

DESCRIPTION

mke2fs.conf is the configuration file for **mke2fs**(8). It controls the default parameters used by **mke2fs**(8) when it is creating ext2, ext3, or ext4 filesystems.

The *mke2fs.conf* file uses an INI-style format. Stanzas, or top-level sections, are delimited by square braces: [ ]. Within each section, each line defines a relation, which assigns tags to values, or to a subsection, which contains further relations or subsections. An example of the INI-style format used by this configuration file follows below:

```
[section1]
        tag1 = value_a
        tag1 = value_b
        tag2 = value_c

[section 2]
        tag3 = {
                subtag1 = subtag_value_a
                subtag1 = subtag_value_b
                subtag2 = subtag_value_c
        }
        tag1 = value_d
        tag2 = value_e
}
```

Comments are delimited by a semicolon (';') or a hash ('#') character at the beginning of the comment, and are terminated by the end of line character.

Tags and values must be quoted using double quotes if they contain spaces. Within a quoted string, the standard backslash interpretations apply: "\n" (for the newline character), "\t" (for the tab character), "\b" (for the backspace character), and "\\" (for the backslash character).

Some relations expect a boolean value. The parser is quite liberal on recognizing "yes", ''y", "true", "t", "1", "on", etc. as a boolean true value, and "no", "n", "false", "nil", "0", "off" as a boolean false value.

The following stanzas are used in the *mke2fs.conf* file. They will be described in more detail in future sections of this document.

[options]
: Contains relations which influence how mke2fs behaves.

[defaults]
: Contains relations which define the default parameters used by **mke2fs**(8). In general, these defaults may be overridden by a definition in the **fs_types** stanza, or by a command-line option provided by the user.

[fs_types]
: Contains relations which define defaults that should be used for specific file system and usage types. The file system type and usage type can be specified explicitly using the **−t**and**−T** options to **mke2fs**(8), respectively.

[devices]
: Contains relations which define defaults for specific devices.

THE [options] STANZA

The following relations are defined in the *[options]* stanza.

proceed_delay
: If this relation is set to a positive integer, then mke2fs will wait *proceed_delay* seconds after asking the user for permission to proceed and then continue, even if the user has not answered the

question. Defaults to 0, which means to wait until the user answers the question one way or another.

sync_kludge
If this relation is set to a positive integer, then while writing the inode table, mke2fs will request the operating system flush out pending writes to initialize the inode table every *sync_kludge* block groups. This is needed to work around buggy kernels that don't handle writeback throttling correctly.

THE [defaults] STANZA

The following relations are defined in the *[defaults]* stanza.

fs_type This relation specifies the default filesystem type if the user does not specify it via the **−t** option, or if **mke2fs** is not started using a program name of the form **mkfs.***fs-type*. If both the user and the **mke2fs.conf** file do not specify a default filesystem type, mke2fs will use a default filesystem type of *ext3* if a journal was requested via a command-line option, or *ext2* if not.

undo_dir
This relation specifies the directory where the undo file should be stored. It can be overridden via the **E2FSPROGS_UNDO_DIR** environment variable. If the directory location is set to the value *none*, **mke2fs** will not create an undo file.

In addition, any tags that can be specified in a per-file system tags subsection as defined below (e.g., *blocksize*, *hash_alg*, *inode_ratio*, *inode_size*, *reserved_ratio*, etc.) can also be specified in the *defaults* stanza to specify the default value to be used if the user does not specify one on the command line, and the filesystem-type specific section of the configuration file does not specify a default value.

THE [fs_types] STANZA

Each tag in the *[fs_types]* stanza names a filesystem type or usage type which can be specified via the **−t** or **−T** options to **mke2fs**(8), respectively.

The **mke2fs** program constructs a list of fs_types by concatenating the filesystem type (i.e., ext2, ext3, etc.) with the usage type list. For most configuration options, **mke2fs** will look for a subsection in the *[fs_types]* stanza corresponding with each entry in the constructed list, with later entries overriding earlier filesystem or usage types. For example, consider the following **mke2fs.conf** fragment:

```
[defaults]
        base_features = sparse_super,filetype,resize_inode,dir_index
        blocksize = 4096
        inode_size = 256
        inode_ratio = 16384

[fs_types]
        ext3 = {
                features = has_journal
        }
        ext4 = {
                features = extents,flex_bg
                inode_size = 256
        }
        small = {
                blocksize = 1024
                inode_ratio = 4096
        }
        floppy = {
                features = ^resize_inode
                blocksize = 1024
                inode_size = 128
        }
```

If mke2fs started with a program name of **mke2fs.ext4**, then the filesystem type of ext4 will be used. If the filesystem is smaller than 3 megabytes, and no usage type is specified, then **mke2fs** will use a default usage type of *floppy*. This results in an fs_types list of "ext4, floppy". Both the ext4 subsection and the floppy subsection define an *inode_size* relation, but since the later entries in the fs_types list supersede earlier ones, the configuration parameter for fs_types.floppy.inode_size will be used, so the filesystem will have an inode size of 128.

The exception to this resolution is the *features* tag, which specifies a set of changes to the features used by the filesystem, and which is cumulative. So in the above example, first the configuration relation defaults.base_features would enable an initial feature set with the sparse_super, filetype, resize_inode, and dir_index features enabled. Then configuration relation fs_types.ext4.features would enable the extents and flex_bg features, and finally the configuration relation fs_types.floppy.features would remove the resize_inode feature, resulting in a filesystem feature set consisting of the sparse_super, filetype, dir_index, extents_and flex_bg features.

For each filesystem type, the following tags may be used in that fs_type's subsection. These tags may also be used in the *default* section:

base_features
: This relation specifies the features which are initially enabled for this filesystem type. Only one *base_features* will be used, so if there are multiple entries in the fs_types list whose subsections define the *base_features* relation, only the last will be used by **mke2fs**(8).

enable_periodic_fsck
: This boolean relation specifies whether periodic filesystem checks should be enforced at boot time. If set to true, checks will be forced every 180 days, or after a random number of mounts. These values may be changed later via the **-i** and **-c** command-line options to **tune2fs**(8).

errors
: 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. *errors* can be one of the following:

continue	Continue normal execution.
remount-ro	Remount filesystem read-only.
panic	Cause a kernel panic.

features
: This relation specifies a comma-separated list of features edit requests which modify the feature set used by the newly constructed filesystem. The syntax is the same as the **-O** command-line option to **mke2fs**(8); that is, a feature can be prefixed by a caret ('^') symbol to disable a named feature. Each *feature* relation specified in the fs_types list will be applied in the order found in the fs_types list.

force_undo
: This boolean relation, if set to a value of true, forces **mke2fs** to always try to create an undo file, even if the undo file might be huge and it might extend the time to create the filesystem image because the inode table isn't being initialized lazily.

default_features
: This relation specifies set of features which should be enabled or disabled after applying the features listed in the *base_features* and *features* relations. It may be overridden by the **-O** command-line option to **mke2fs**(8).

auto_64-bit_support
: This relation is a boolean which specifies whether **mke2fs**(8) should automatically add the 64bit feature if the number of blocks for the file system requires this feature to be enabled. The resize_inode feature is also automatically disabled since it doesn't support 64-bit block numbers.

default_mntopts
: This relation specifies the set of mount options which should be enabled by default. These may be changed at a later time with the **-o** command-line option to **tune2fs**(8).

blocksize
This relation specifies the default blocksize if the user does not specify a blocksize on the command line.

lazy_itable_init
This boolean relation specifies whether the inode table should be lazily initialized. It only has meaning if the uninit_bg feature is enabled. If lazy_itable_init is true and the uninit_bg feature is enabled, the inode table will not be fully initialized by **mke2fs**(8). 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.

lazy_journal_init
This boolean relation specifies whether the journal inode should be lazily initialized. It only has meaning if the has_journal feature is enabled. If lazy_journal_init is true, 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.

journal_location
This relation specifies the location of the journal.

num_backup_sb
This relation indicates whether file systems with the **sparse_super2** feature enabled should be created with 0, 1, or 2 backup superblocks.

packed_meta_blocks
This boolean relation specifies whether the allocation bitmaps, inode table, and journal should be located at the beginning of the file system.

inode_ratio
This relation specifies the default inode ratio if the user does not specify one on the command line.

inode_size
This relation specifies the default inode size if the user does not specify one on the command line.

reserved_ratio
This relation specifies the default percentage of filesystem blocks reserved for the super-user, if the user does not specify one on the command line.

hash_alg
This relation specifies the default hash algorithm used for the new filesystems with hashed b-tree directories. Valid algorithms accepted are: *legacy*, *half_md4*, and *tea*.

flex_bg_size
This relation specifies the number of block groups that will be packed together to create one large virtual block group on 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.

options This relation specifies additional extended options which should be treated by **mke2fs**(8) as if they were prepended to the argument of the **-E** option. This can be used to configure the default extended options used by **mke2fs**(8) on a per-filesystem type basis.

discard This boolean relation specifies whether the **mke2fs**(8) should attempt to discard device prior to filesystem creation.

cluster_size
This relation specifies the default cluster size if the bigalloc file system feature is enabled. It can be overridden via the **−C** command line option to **mke2fs**(8)

make_hugefiles
This boolean relation enables the creation of pre-allocated files as part of formatting the file system. The extent tree blocks for these pre-allocated files will be placed near the beginning of the file system, so that if all of the other metadata blocks are also configured to be placed near the

beginning of the file system (by disabling the backup superblocks, using the packed_meta_blocks option, etc.), the data blocks of the pre-allocated files will be contiguous.

hugefiles_dir
This relation specifies the directory where huge files are created, relative to the filesystem root.

hugefiles_uid
This relation controls the user ownership for all of the files and directories created by the *make_hugefiles* feature.

hugefiles_gid
This relation controls the group ownership for all of the files and directories created by the *make_hugefiles* feature.

hugefiles_umask
This relation specifies the umask used when creating the files and directories by the *make_hugefiles* feature.

num_hugefiles
This relation specifies the number of huge files to be created. If this relation is not specified, or is set to zero, and the *hugefiles_size* relation is non-zero, then *make_hugefiles* will create as many huge files as can fit to fill the entire file system.

hugefiles_slack
This relation specifies how much space should be reserved for other files.

hugefiles_size
This relation specifies the size of the huge files. If this relation is not specified, the default is to fill the entire file system.

hugefiles_align
This relation specifies the alignment for the start block of the huge files. It also forces the size of huge files to be a multiple of the requested alignment. If this relation is not specified, no alignment requirement will be imposed on the huge files.

hugefiles_align_disk
This relations specifies whether the alignment should be relative to the beginning of the hard drive (assuming that the starting offset of the partition is available to mke2fs). The default value is false, which will cause hugefile alignment to be relative to the beginning of the file system.

hugefiles_name
This relation specifies the base file name for the huge files.

hugefiles_digits
This relation specifies the (zero-padded) width of the field for the huge file number.

zero_hugefiles
This boolean relation specifies whether or not zero blocks will be written to the hugefiles while **mke2fs**(8) is creating them. By default, zero blocks will be written to the huge files to avoid stale data from being made available to potentially untrusted user programs, unless the device supports a discard/trim operation which will take care of zeroing the device blocks. By setting *zero_hugefiles* to false, this step will always be skipped, which can be useful if it is known that the disk has been previously erased, or if the user programs that will have access to the huge files are trusted to not reveal stale data.

encoding
This relation defines the file name encoding to be used if the casefold feature is enabled. Currently the only valid encoding is utf8-12.1 or utf8, which requests the most recent Unicode version; since 12.1 is the only available Unicode version, utf8 and utf8-12.1 have the same result. *encoding_flags* This relation defines encoding-specific flags. For utf8 encodings, the only available flag is strict, which will cause attempts to create file names containing invalid Unicode characters to be rejected by the kernel. Strict mode is not enabled by default.

THE [devices] STANZA

Each tag in the *[devices]* stanza names device name so that per-device defaults can be specified.

fs_type This relation specifies the default parameter for the **−t** option, if this option isn't specified on the command line.

usage_types
This relation specifies the default parameter for the **−T** option, if this option isn't specified on the command line.

FILES

/etc/mke2fs.conf
The configuration file for **mke2fs**(8).

SEE ALSO

mke2fs(8)