NAME

mkfs.xfs – construct an XFS filesystem

SYNOPSIS

mkfs.xfs [**–b** *block_size_options*] [**–m** *global_metadata_options*] [**–d** *data_section_options*] [**–f**] [**–i** *inode_options*] [**–l** *log_section_options*] [**–n** *naming_options*] [**–p** *protofile*] [**–q**] [**–r** *realtime_section_options*] [**–s** *sector_size_options*] [**–L** *label*] [**–N**] [**–K**] *device*
mkfs.xfs –V

DESCRIPTION

mkfs.xfs constructs an XFS filesystem by writing on a special file using the values found in the arguments of the command line. It is invoked automatically by **mkfs**(8) when it is given the **–t xfs** option.

In its simplest (and most commonly used form), the size of the filesystem is determined from the disk driver. As an example, to make a filesystem with an internal log on the first partition on the first SCSI disk, use:

mkfs.xfs /dev/sda1

The metadata log can be placed on another device to reduce the number of disk seeks. To create a filesystem on the first partition on the first SCSI disk with a 10MiB log located on the first partition on the second SCSI disk, use:

mkfs.xfs –l logdev=/dev/sdb1,size=10m /dev/sda1

Each of the *option* elements in the argument list above can be given as multiple comma-separated suboptions if multiple suboptions apply to the same option. Equivalently, each main option can be given multiple times with different suboptions. For example, **–l internal,size=10m** and **–l internal –l size=10m** are equivalent.

In the descriptions below, sizes are given in sectors, bytes, blocks, kilobytes, megabytes, gigabytes, etc. Sizes are treated as hexadecimal if prefixed by 0x or 0X, octal if prefixed by 0, or decimal otherwise. The following lists possible multiplication suffixes:

s – multiply by sector size (default = 512, see **–s** option below).
b – multiply by filesystem block size (default = 4K, see **–b** option below).
k – multiply by one kilobyte (1,024 bytes).
m – multiply by one megabyte (1,048,576 bytes).
g – multiply by one gigabyte (1,073,741,824 bytes).
t – multiply by one terabyte (1,099,511,627,776 bytes).
p – multiply by one petabyte (1,024 terabytes).
e – multiply by one exabyte (1,048,576 terabytes).

When specifying parameters in units of sectors or filesystem blocks, the **–s** option or the **–b** option may be used to specify the size of the sector or block. If the size of the block or sector is not specified, the default sizes (block: 4KiB, sector: 512B) will be used.

Many feature options allow an optional argument of 0 or 1, to explicitly disable or enable the functionality.

OPTIONS

–b *block_size_options*

This option specifies the fundamental block size of the filesystem. The valid *block_size_option* is:

size=*value*

The filesystem block size is specified with a *value* in bytes. The default value is 4096 bytes (4 KiB), the minimum is 512, and the maximum is 65536 (64 KiB).

Although **mkfs.xfs** will accept any of these values and create a valid filesystem, XFS on Linux can only mount filesystems with pagesize or smaller blocks.

−m *global_metadata_options*

These options specify metadata format options that either apply to the entire filesystem or aren't easily characterised by a specific functionality group. The valid *global_metadata_options* are:

crc=*value*

This is used to create a filesystem which maintains and checks CRC information in all metadata objects on disk. The value is either 0 to disable the feature, or 1 to enable the use of CRCs.

CRCs enable enhanced error detection due to hardware issues, whilst the format changes also improves crash recovery algorithms and the ability of various tools to validate and repair metadata corruptions when they are found. The CRC algorithm used is CRC32c, so the overhead is dependent on CPU architecture as some CPUs have hardware acceleration of this algorithm. Typically the overhead of calculating and checking the CRCs is not noticeable in normal operation.

By default, **mkfs.xfs** will enable metadata CRCs.

finobt=*value*

This option enables the use of a separate free inode btree index in each allocation group. The value is either 0 to disable the feature, or 1 to create a free inode btree in each allocation group.

The free inode btree mirrors the existing allocated inode btree index which indexes both used and free inodes. The free inode btree does not index used inodes, allowing faster, more consistent inode allocation performance as filesystems age.

By default, **mkfs.xfs** will create free inode btrees for filesystems created with the (default) **−m crc=1** option set. When the option **−m crc=0** is used, the free inode btree feature is not supported and is disabled.

uuid=*value*

Use the given value as the filesystem UUID for the newly created filesystem. The default is to generate a random UUID.

rmapbt=*value*

This option enables the creation of a reverse-mapping btree index in each allocation group. The value is either 0 to disable the feature, or 1 to create the btree.

The reverse mapping btree maps filesystem blocks to the owner of the filesystem block. Most of the mappings will be to an inode number and an offset, though there will also be mappings to filesystem metadata. This secondary metadata can be used to validate the primary metadata or to pinpoint exactly which data has been lost when a disk error occurs.

By default, **mkfs.xfs** will not create reverse mapping btrees. This feature is only available for filesystems created with the (default) **−m crc=1** option set. When the option **−m crc=0** is used, the reverse mapping btree feature is not supported and is disabled.

reflink=*value*

This option enables the use of a separate reference count btree index in each allocation group. The value is either 0 to disable the feature, or 1 to create a reference count btree in each allocation group.

The reference count btree enables the sharing of physical extents between the data forks of different files, which is commonly known as "reflink". Unlike traditional Unix filesystems which assume that every inode and logical block pair map to a unique physical block, a reflink-capable XFS filesystem

removes the uniqueness requirement, allowing up to four billion arbitrary inode/logical block pairs to map to a physical block. If a program tries to write to a multiply-referenced block in a file, the write will be redirected to a new block, and that file's logical-to-physical mapping will be changed to the new block ("copy on write"). This feature enables the creation of per-file snapshots and deduplication. It is only available for the data forks of regular files.

By default, **mkfs.xfs** will create reference count btrees and therefore will enable the reflink feature. This feature is only available for filesystems created with the (default) **−m crc=1** option set. When the option **−m crc=0** is used, the reference count btree feature is not supported and reflink is disabled.

Note: the filesystem DAX mount option (**−o dax**) is incompatible with reflink-enabled XFS filesystems. To use filesystem DAX with XFS, specify the **−m reflink=0** option to mkfs.xfs to disable the reflink feature.

−d *data_section_options*

These options specify the location, size, and other parameters of the data section of the filesystem. The valid *data_section_options* are:

agcount=*value*

This is used to specify the number of allocation groups. The data section of the filesystem is divided into allocation groups to improve the performance of XFS. More allocation groups imply that more parallelism can be achieved when allocating blocks and inodes. The minimum allocation group size is 16 MiB; the maximum size is just under 1 TiB. The data section of the filesystem is divided into *value* allocation groups (default value is scaled automatically based on the underlying device size).

agsize=*value*

This is an alternative to using the **agcount** suboption. The *value* is the desired size of the allocation group expressed in bytes (usually using the **m** or **g** suffixes). This value must be a multiple of the filesystem block size, and must be at least 16MiB, and no more than 1TiB, and may be automatically adjusted to properly align with the stripe geometry. The **agcount** and **agsize** suboptions are mutually exclusive.

cowextsize=*value*

Set the copy-on-write extent size hint on all inodes created by **mkfs.xfs**. The value must be provided in units of filesystem blocks. If the value is zero, the default value (currently 32 blocks) will be used. Directories will pass on this hint to newly created children.

name=*value*

This can be used to specify the name of the special file containing the filesystem. In this case, the log section must be specified as **internal** (with a size, see the **−l** option below) and there can be no real-time section.

file[=*value***]**

This is used to specify that the file given by the **name** suboption is a regular file. The *value* is either 0 or 1, with 1 signifying that the file is regular. This suboption is used only to make a filesystem image. If the *value* is omitted then 1 is assumed.

size=*value*

This is used to specify the size of the data section. This suboption is required if **−d file[=1]** is given. Otherwise, it is only needed if the filesystem should occupy less space than the size of the special file.

sunit=*value*
This is used to specify the stripe unit for a RAID device or a logical volume. The *value* has to be specified in 512-byte block units. Use the **su** suboption to specify the stripe unit size in bytes. This suboption ensures that data allocations will be stripe unit aligned when the current end of file is being extended and the file size is larger than 512KiB. Also inode allocations and the internal log will be stripe unit aligned.

su=*value*
This is an alternative to using **sunit.** The **su** suboption is used to specify the stripe unit for a RAID device or a striped logical volume. The *value* has to be specified in bytes, (usually using the **m** or **g** suffixes). This *value* must be a multiple of the filesystem block size.

swidth=*value*
This is used to specify the stripe width for a RAID device or a striped logical volume. The *value* has to be specified in 512-byte block units. Use the **sw** suboption to specify the stripe width size in bytes. This suboption is required if **–d sunit** has been specified and it has to be a multiple of the **–d sunit** suboption.

sw=*value*
suboption is an alternative to using **swidth.** The **sw** suboption is used to specify the stripe width for a RAID device or striped logical volume. The *value* is expressed as a multiplier of the stripe unit, usually the same as the number of stripe members in the logical volume configuration, or data disks in a RAID device.

When a filesystem is created on a logical volume device, **mkfs.xfs** will automatically query the logical volume for appropriate **sunit** and **swidth** values.

noalign
This option disables automatic geometry detection and creates the filesystem without stripe geometry alignment even if the underlying storage device provides this information.

rtinherit=*value*
If set, all inodes created by **mkfs.xfs** will be created with the realtime flag set. Directories will pass on this flag to newly created children.

projinherit=*value*
All inodes created by **mkfs.xfs** will be assigned this project quota id. Directories will pass on the project id to newly created children.

extszinherit=*value*
All inodes created by **mkfs.xfs** will have this extent size hint applied. The value must be provided in units of filesystem blocks. Directories will pass on this hint to newly created children.

–f Force overwrite when an existing filesystem is detected on the device. By default, **mkfs.xfs** will not write to the device if it suspects that there is a filesystem or partition table on the device already.

–i *inode_options*
This option specifies the inode size of the filesystem, and other inode allocation parameters. The XFS inode contains a fixed-size part and a variable-size part. The variable-size part, whose size is affected by this option, can contain: directory data, for small directories; attribute data, for small attribute sets; symbolic link data, for small symbolic links; the extent list for the file, for files with a small number of extents; and the root of a tree describing the location of extents for the file, for files with a large number of extents.

The valid *inode_options* are:

size=*value* | **perblock=***value*

The inode size is specified either as a *value* in bytes with **size=** or as the number fitting in a filesystem block with **perblock=**. The minimum (and default) *value* is 256 bytes without crc, 512 bytes with crc enabled. The maximum *value* is 2048 (2 KiB) subject to the restriction that the inode size cannot exceed one half of the filesystem block size.

XFS uses 64-bit inode numbers internally; however, the number of significant bits in an inode number is affected by filesystem geometry. In practice, filesystem size and inode size are the predominant factors. The Linux kernel (on 32 bit hardware platforms) and most applications cannot currently handle inode numbers greater than 32 significant bits, so if no inode size is given on the command line, **mkfs.xfs** will attempt to choose a size such that inode numbers will be < 32 bits. If an inode size is specified, or if a filesystem is sufficiently large, **mkfs.xfs** will warn if this will create inode numbers > 32 significant bits.

maxpct=*value*

This specifies the maximum percentage of space in the filesystem that can be allocated to inodes. The default *value* is 25% for filesystems under 1TB, 5% for filesystems under 50TB and 1% for filesystems over 50TB.

In the default inode allocation mode, inode blocks are chosen such that inode numbers will not exceed 32 bits, which restricts the inode blocks to the lower portion of the filesystem. The data block allocator will avoid these low blocks to accommodate the specified maxpct, so a high value may result in a filesystem with nothing but inodes in a significant portion of the lower blocks of the filesystem. (This restriction is not present when the filesystem is mounted with the *inode64* option on 64-bit platforms).

Setting the value to 0 means that essentially all of the filesystem can become inode blocks, subject to inode32 restrictions.

This value can be modified with *xfs_growfs(8)*.

align[=*value***]**

This is used to specify that inode allocation is or is not aligned. The *value* is either 0 or 1, with 1 signifying that inodes are allocated aligned. If the *value* is omitted, 1 is assumed. The default is that inodes are aligned. Aligned inode access is normally more efficient than unaligned access; alignment must be established at the time the filesystem is created, since inodes are allocated at that time. This option can be used to turn off inode alignment when the filesystem needs to be mountable by a version of IRIX that does not have the inode alignment feature (any release of IRIX before 6.2, and IRIX 6.2 without XFS patches).

attr=*value*

This is used to specify the version of extended attribute inline allocation policy to be used. By default, this is 2, which uses an efficient algorithm for managing the available inline inode space between attribute and extent data.

The previous version 1, which has fixed regions for attribute and extent data, is kept for backwards compatibility with kernels older than version 2.6.16.

projid32bit[=*value***]**

This is used to enable 32bit quota project identifiers. The *value* is either 0 or 1, with 1 signifying that 32bit projid are to be enabled. If the value is omitted, 1 is assumed. (This default changed in release version 3.2.0.)

sparse[=*value***]**
Enable sparse inode chunk allocation. The *value* is either 0 or 1, with 1 signifying that sparse allocation is enabled. If the value is omitted, 1 is assumed. Sparse inode allocation is disabled by default. This feature is only available for filesystems formatted with **−m crc=1.**

When enabled, sparse inode allocation allows the filesystem to allocate smaller than the standard 64-inode chunk when free space is severely limited. This feature is useful for filesystems that might fragment free space over time such that no free extents are large enough to accommodate a chunk of 64 inodes. Without this feature enabled, inode allocations can fail with out of space errors under severe fragmented free space conditions.

−l *log_section_options*
These options specify the location, size, and other parameters of the log section of the filesystem. The valid *log_section_options* are:

agnum=*value*
If the log is internal, allocate it in this AG.

internal[=*value***]**
This is used to specify that the log section is a piece of the data section instead of being another device or logical volume. The *value* is either 0 or 1, with 1 signifying that the log is internal. If the *value* is omitted, 1 is assumed.

logdev=*device*
This is used to specify that the log section should reside on the *device* separate from the data section. The **internal=1** and **logdev** options are mutually exclusive.

size=*value*
This is used to specify the size of the log section.

If the log is contained within the data section and **size** isn't specified, **mkfs.xfs** will try to select a suitable log size depending on the size of the filesystem. The actual logsize depends on the filesystem block size and the directory block size.

Otherwise, the **size** suboption is only needed if the log section of the filesystem should occupy less space than the size of the special file. The *value* is specified in bytes or blocks, with a **b** suffix meaning multiplication by the filesystem block size, as described above. The overriding minimum value for size is 512 blocks. With some combinations of filesystem block size, inode size, and directory block size, the minimum log size is larger than 512 blocks.

version=*value*
This specifies the version of the log. The current default is 2, which allows for larger log buffer sizes, as well as supporting stripe-aligned log writes (see the sunit and su options, below).

The previous version 1, which is limited to 32k log buffers and does not support stripe-aligned writes, is kept for backwards compatibility with very old 2.4 kernels.

sunit=*value*
This specifies the alignment to be used for log writes. The *value* has to be specified in 512-byte block units. Use the **su** suboption to specify the log stripe unit size in bytes. Log writes will be aligned on this boundary, and rounded up to this boundary. This gives major improvements in performance

on some configurations such as software RAID5 when the **sunit** is specified as the filesystem block size. The equivalent byte value must be a multiple of the filesystem block size. Version 2 logs are automatically selected if the log **sunit** suboption is specified.

The **su** suboption is an alternative to using **sunit.**

su=*value*
This is used to specify the log stripe. The *value* has to be specified in bytes, (usually using the **s** or **b** suffixes). This value must be a multiple of the filesystem block size. Version 2 logs are automatically selected if the log **su** suboption is specified.

lazy-count=*value*
This changes the method of logging various persistent counters in the superblock. Under metadata intensive workloads, these counters are updated and logged frequently enough that the superblock updates become a serialization point in the filesystem. The *value* can be either 0 or 1.

With **lazy-count=1**, the superblock is not modified or logged on every change of the persistent counters. Instead, enough information is kept in other parts of the filesystem to be able to maintain the persistent counter values without needed to keep them in the superblock. This gives significant improvements in performance on some configurations. The default *value* is 1 (on) so you must specify **lazy-count=0** if you want to disable this feature for older kernels which don't support it.

−n *naming_options*
These options specify the version and size parameters for the naming (directory) area of the filesystem. The valid *naming_options* are:

size=*value*
The directory block size is specified with a *value* in bytes. The block size must be a power of 2 and cannot be less than the filesystem block size. The default size *value* for version 2 directories is 4096 bytes (4 KiB), unless the filesystem block size is larger than 4096, in which case the default *value* is the filesystem block size. For version 1 directories the block size is the same as the filesystem block size.

version=*value*
The naming (directory) version *value* can be either 2 or 'ci', defaulting to 2 if unspecified. With version 2 directories, the directory block size can be any power of 2 size from the filesystem block size up to 65536.

The **version=ci** option enables ASCII only case-insensitive filename lookup and version 2 directories. Filenames are case-preserving, that is, the names are stored in directories using the case they were created with.

Note: Version 1 directories are not supported.

ftype=*value*
This feature allows the inode type to be stored in the directory structure so that the **readdir**(3) and **getdents**(2) do not need to look up the inode to determine the inode type.

The *value* is either 0 or 1, with 1 signifying that filetype information will be stored in the directory structure. The default value is 1.

When CRCs are enabled (the default), the ftype functionality is always enabled, and cannot be turned off.

−p *protofile*

If the optional **−p** *protofile* argument is given, **mkfs.xfs** uses *protofile* as a prototype file and takes its directions from that file. The blocks and inodes specifiers in the *protofile* are provided for backwards compatibility, but are otherwise unused. The syntax of the protofile is defined by a number of tokens separated by spaces or newlines. Note that the line numbers are not part of the syntax but are meant to help you in the following discussion of the file contents.

```
1       /stand/diskboot
2       4872 110
3       d--777 3 1
4       usr     d--777 3 1
5       sh      ---755 3 1 /bin/sh
6       ken     d--755 6 1
7               $
8       b0      b--644 3 1 0 0
9       c0      c--644 3 1 0 0
10      fifo    p--644 3 1
11      slink   l--644 3 1 /a/symbolic/link
12      :  This is a comment line
13      $
14      $
```

Line 1 is a dummy string. (It was formerly the bootfilename.) It is present for backward compatibility; boot blocks are not used on SGI systems.

Note that some string of characters must be present as the first line of the proto file to cause it to be parsed correctly; the value of this string is immaterial since it is ignored.

Line 2 contains two numeric values (formerly the numbers of blocks and inodes). These are also merely for backward compatibility: two numeric values must appear at this point for the proto file to be correctly parsed, but their values are immaterial since they are ignored.

The lines 3 through 11 specify the files and directories you want to include in this filesystem. Line 3 defines the root directory. Other directories and files that you want in the filesystem are indicated by lines 4 through 6 and lines 8 through 10. Line 11 contains symbolic link syntax.

Notice the dollar sign (**$**) syntax on line 7. This syntax directs the **mkfs.xfs** command to terminate the branch of the filesystem it is currently on and then continue from the directory specified by the next line, in this case line 8. It must be the last character on a line. The colon on line 12 introduces a comment; all characters up until the following newline are ignored. Note that this means you cannot have a file in a prototype file whose name contains a colon. The **$** on lines 13 and 14 end the process, since no additional specifications follow.

File specifications provide the following:

* file mode
* user ID
* group ID
* the file's beginning contents

A 6-character string defines the mode for a file. The first character of this string defines the file type. The character range for this first character is **−bcdpl.** A file may be a regular file, a block special file, a character special file, directory files, named pipes (first-in, first out files), and symbolic links. The second character of the mode string is used to specify setuserID mode, in which case it is **u**. If setuserID mode is not specified, the second character is **−**. The third character of the mode string is used to specify the setgroupID mode, in which case it is **g**. If setgroupID mode is not specified, the third character is **−**. The remaining characters of the mode string are a three digit octal number. This octal number defines the owner, group, and other read, write, and execute permissions for the file, respectively. For more information on file permissions, see the **chmod**(1)

command.

Following the mode character string are two decimal number tokens that specify the user and group IDs of the file's owner.

In a regular file, the next token specifies the pathname from which the contents and size of the file are copied. In a block or character special file, the next token are two decimal numbers that specify the major and minor device numbers. When a file is a symbolic link, the next token specifies the contents of the link.

When the file is a directory, the **mkfs.xfs** command creates the entries **dot** (.) and **dot-dot** (..) and then reads the list of names and file specifications in a recursive manner for all of the entries in the directory. A scan of the protofile is always terminated with the dollar (**$**) token.

−q Quiet option. Normally **mkfs.xfs** prints the parameters of the filesystem to be constructed; the **−q** flag suppresses this.

−r *realtime_section_options*
These options specify the location, size, and other parameters of the real-time section of the filesystem. The valid *realtime_section_options* are:

rtdev=*device*
This is used to specify the *device* which should contain the real-time section of the filesystem. The suboption value is the name of a block device.

extsize=*value*
This is used to specify the size of the blocks in the real-time section of the filesystem. This *value* must be a multiple of the filesystem block size. The minimum allowed size is the filesystem block size or 4 KiB (whichever is larger); the default size is the stripe width for striped volumes or 64 KiB for non-striped volumes; the maximum allowed size is 1 GiB. The real-time extent size should be carefully chosen to match the parameters of the physical media used.

size=*value*
This is used to specify the size of the real-time section. This suboption is only needed if the real-time section of the filesystem should occupy less space than the size of the partition or logical volume containing the section.

noalign
This option disables stripe size detection, enforcing a realtime device with no stripe geometry.

−s *sector_size_options*
This option specifies the fundamental sector size of the filesystem. The valid *sector_size_option* is:

size=*value*
The sector size is specified with a *value* in bytes. The default *sector_size* is 512 bytes. The minimum value for sector size is 512; the maximum is 32768 (32 KiB). The *sector_size* must be a power of 2 size and cannot be made larger than the filesystem block size.

−L *label*
Set the filesystem *label*. XFS filesystem labels can be at most 12 characters long; if *label* is longer than 12 characters, **mkfs.xfs** will not proceed with creating the filesystem. Refer to the **mount**(8) and **xfs_admin**(8) manual entries for additional information.

−N Causes the file system parameters to be printed out without really creating the file system.

−K Do not attempt to discard blocks at mkfs time.

–V Prints the version number and exits.

SEE ALSO

xfs(5), **mkfs**(8), **mount**(8), **xfs_info**(8), **xfs_admin**(8).

BUGS

With a prototype file, it is not possible to specify hard links.