Rocky Enterprise Linux 9.2 Manual Pages on command 'xfs.5'

C:\>man xfs.5

xfs(5) File Formats Manual xfs(5)

NAME

xfs - layout, mount options, and supported file attributes for the XFS filesystem

DESCRIPTION

An XFS filesystem can reside on a regular disk partition or on a logical volume. An XFS filesystem has up to three parts: a data section, a log section, and a real? time section. Using the default mkfs.xfs(8) options, the realtime section is ab? sent, and the log area is contained within the data section. The log section can be either separate from the data section or contained within it. The filesystem sections are divided into a certain number of blocks, whose size is specified at mkfs.xfs(8) time with the -b option.

The data section contains all the filesystem metadata (inodes, directories, indi? rect blocks) as well as the user file data for ordinary (non-realtime) files and the log area if the log is internal to the data section. The data section is di? vided into a number of allocation groups. The number and size of the allocation groups are chosen by mkfs.xfs(8) so that there is normally a small number of equal-sized groups. The number of allocation groups controls the amount of parallelism available in file and block allocation. It should be increased from the default if there is sufficient memory and a lot of allocation activity. The number of alloca? tion groups should not be set very high, since this can cause large amounts of CPU time to be used by the filesystem, especially when the filesystem is nearly full. More allocation groups are added (of the original size) when xfs_growfs(8) is run.

The log section (or area, if it is internal to the data section) is used to store changes to filesystem metadata while the filesystem is running until those changes are made to the data section. It is written sequentially during normal operation and read only during mount. When mounting a filesystem after a crash, the log is read to complete operations that were in progress at the time of the crash.

The realtime section is used to store the data of realtime files. These files had an attribute bit set through xfsctl(3) after file creation, before any data was written to the file. The realtime section is divided into a number of extents of fixed size (specified at mkfs.xfs(8) time). Each file in the realtime section has an extent size that is a multiple of the realtime section extent size.

Each allocation group contains several data structures. The first sector contains the superblock. For allocation groups after the first, the superblock is just a copy and is not updated after mkfs.xfs(8). The next three sectors contain informa? tion for block and inode allocation within the allocation group. Also contained within each allocation group are data structures to locate free blocks and inodes; these are located through the header structures.

Each XFS filesystem is labeled with a Universal Unique Identifier (UUID). The UUID is stored in every allocation group header and is used to help distinguish one XFS filesystem from another, therefore you should avoid using dd(1) or other block-by-block copying programs to copy XFS filesystems. If two XFS filesystems on the same machine have the same UUID, xfsdump(8) may become confused when doing incremental and resumed dumps. xfsdump(8) and xfsrestore(8) are recommended for making copies of XFS filesystems.

OPERATIONS

Some functionality specific to the XFS filesystem is accessible to applications through the xfsctl(3) and by-handle (see open_by_handle(3)) interfaces.

MOUNT OPTIONS

The following XFS-specific mount options may be used when mounting an XFS filesys? tem. Other generic options may be used as well; refer to the mount(8) manual page for more details.

allocsize=size

Sets the buffered I/O end-of-file preallocation size when doing delayed al? location writeout. Valid values for this option are page size (typically

4KiB) through to 1GiB, inclusive, in power-of-2 increments.

The default behavior is for dynamic end-of-file preallocation size, which uses a set of heuristics to optimise the preallocation size based on the current allocation patterns within the file and the access patterns to the file. Specifying a fixed allocsize value turns off the dynamic behavior.

attr2|noattr2

The options enable/disable an "opportunistic" improvement to be made in the way inline extended attributes are stored on-disk. When the new form is used for the first time when attr2 is selected (either when setting or re? moving extended attributes) the on-disk superblock feature bit field will be updated to reflect this format being in use.

The default behavior is determined by the on-disk feature bit indicating that attr2 behavior is active. If either mount option it set, then that be? comes the new default used by the filesystem.

CRC enabled filesystems always use the attr2 format, and so will reject the noattr2 mount option if it is set.

discard|nodiscard

Enable/disable the issuing of commands to let the block device reclaim space freed by the filesystem. This is useful for SSD devices, thinly provisioned LUNs and virtual machine images, but may have a performance impact.

Note: It is currently recommended that you use the fstrim application to discard unused blocks rather than the discard mount option because the per? formance impact of this option is quite severe. For this reason, nodiscard is the default.

grpid|bsdgroups|nogrpid|sysvgroups

These options define what group ID a newly created file gets. When grpid is set, it takes the group ID of the directory in which it is created; other? wise it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

filestreams

Make the data allocator use the filestreams allocation mode across the en? tire filesystem rather than just on directories configured to use it.

ikeep|noikeep

When ikeep is specified, XFS does not delete empty inode clusters and keeps them around on disk. When noikeep is specified, empty inode clusters are returned to the free space pool. noikeep is the default.

inode32|inode64

When inode32 is specified, it indicates that XFS limits inode creation to locations which will not result in inode numbers with more than 32 bits of significance.

When inode64 is specified, it indicates that XFS is allowed to create inodes at any location in the filesystem, including those which will result in in? ode numbers occupying more than 32 bits of significance.

inode32 is provided for backwards compatibility with older systems and ap? plications, since 64 bits inode numbers might cause problems for some appli? cations that cannot handle large inode numbers. If applications are in use which do not handle inode numbers bigger than 32 bits, the inode32 option should be specified.

For kernel v3.7 and later, inode64 is the default.

largeio|nolargeio

If "nolargeio" is specified, the optimal I/O reported in st_blksize by stat(2) will be as small as possible to allow user applications to avoid in? efficient read/modify/write I/O. This is typically the page size of the ma? chine, as this is the granularity of the page cache.

If "largeio" specified, a filesystem that was created with a "swidth" speci? fied will return the "swidth" value (in bytes) in st_blksize. If the filesystem does not have a "swidth" specified but does specify an "alloc? size" then "allocsize" (in bytes) will be returned instead. Otherwise the behavior is the same as if "nolargeio" was specified. nolargeio is the de? fault.

logbufs=value

Set the number of in-memory log buffers. Valid numbers range from 2?8 in? clusive.

The default value is 8 buffers.

If the memory cost of 8 log buffers is too high on small systems, then it

may be reduced at some cost to performance on metadata intensive workloads. The logbsize option below controls the size of each buffer and so is also relevant to this case.

logbsize=value

Set the size of each in-memory log buffer. The size may be specified in bytes, or in kibibytes (KiB) with a "k" suffix. Valid sizes for version 1 and version 2 logs are 16384 (value=16k) and 32768 (value=32k). Valid sizes for version 2 logs also include 65536 (value=64k), 131072 (value=128k) and 262144 (value=256k). The logbsize must be an integer multiple of the log stripe unit configured at mkfs time.

The default value for version 1 logs is 32768, while the default value for version 2 logs is max(32768, log_sunit).

logdev=device and rtdev=device

Use an external log (metadata journal) and/or real-time device. An XFS filesystem has up to three parts: a data section, a log section, and a real-time section. The real-time section is optional, and the log section can be separate from the data section or contained within it.

noalign

Data allocations will not be aligned at stripe unit boundaries. This is only relevant to filesystems created with non-zero data alignment parameters (sunit, swidth) by mkfs.

norecovery

The filesystem will be mounted without running log recovery. If the filesystem was not cleanly unmounted, it is likely to be inconsistent when mounted in "norecovery" mode. Some files or directories may not be accessi? ble because of this. Filesystems mounted "norecovery" must be mounted read-only or the mount will fail.

nouuid Don't check for double mounted file systems using the file system uuid. This is useful to mount LVM snapshot volumes, and often used in combination with "norecovery" for mounting read-only snapshots.

noquota

Forcibly turns off all quota accounting and enforcement within the filesys? tem.

uquota/usrquota/quota/uqnoenforce/qnoenforce

User disk quota accounting enabled, and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

gquota/grpquota/gqnoenforce

Group disk quota accounting enabled and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

pquota/prjquota/pqnoenforce

Project disk quota accounting enabled and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

sunit=value and swidth=value

Used to specify the stripe unit and width for a RAID device or a stripe volume. "value" must be specified in 512-byte block units. These options are only relevant to filesystems that were created with non-zero data alignment parameters.

The sunit and swidth parameters specified must be compatible with the existing filesystem alignment characteristics. In general, that means the only valid changes to sunit are increasing it by a power-of-2 multiple. Valid swidth values are any integer multiple of a valid sunit value.

Typically the only time these mount options are necessary if after an underlying RAID device has had it's geometry modified, such as adding a new disk to a RAID5 lun and reshaping it.

swalloc

Data allocations will be rounded up to stripe width boundaries when the current end of file is being extended and the file size is larger than the stripe width size.

wsync When specified, all filesystem namespace operations are executed synchronously. This ensures that when the namespace operation (create, unlink, etc) completes, the change to the namespace is on stable storage. This is useful in HA setups where failover must not result in clients seeing inconsistent namespace presentation during or after a failover event.

REMOVED MOUNT OPTIONS

The following mount options have been removed from the kernel, and will yield mount failures if specified. Mount options are deprecated for a significant period time

prior to removal.

Name	Removed
delaylog/nodelaylog	v4.0
ihashsize	v4.0
irixsgid	v4.0
osyncisdsync/osyncisosync	v4.0
barrier/nobarrier	v4.19

FILE ATTRIBUTES

The XFS filesystem supports setting the following file attributes on Linux systems using the chattr(1) utility:

a - append only

A - no atime updates

d - no dump

i - immutable

S - synchronous updates

For descriptions of these attribute flags, please refer to the chattr(1) man page.

SEE ALSO

chattr(1), xfsctl(3), mount(8), mkfs.xfs(8), xfs_info(8), xfs_admin(8), xfsdump(8), xfsrestore(8).