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

***C:\>man xfs_io.8***

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

NAME

xfs_io - debug the I/O path of an XFS filesystem

SYNOPSIS

xfs_io [ -adfimrRstxT ] [ -c cmd ] ... [ -C cmd ] ... [ -p prog ] [ file ]

xfs_io -V

DESCRIPTION

xfs_io is a debugging tool like xfs_db(8), but is aimed at examining the regular file I/O paths rather than the raw XFS volume itself. These code paths include not only the obvious read/write/mmap interfaces for manipulating files, but also cover all of the XFS extensions (such as space preallocation, additional inode flags, etc).

OPTIONS

xfs_io commands may be run interactively (the default) or as arguments on the com? mand line. Interactive mode always runs commands on the current open file, whilst commands run from the command line may be repeated on all open files rather than just the current open file. In general, open file iteration will occur for com? mands that operate on file content or state. In contrast, commands that operate on filesystem or system-wide state will only be run on the current file regardless of how many files are currently open. Multiple arguments may be given on the command line and they are run in the sequence given. The program exits one all commands have been run.

-c cmd Run the specified command on all currently open files. To maintain com?
patibility with historical usage, commands that can not be run on all
open files will still be run but only execute once on the current open
file. Multiple -c arguments may be given and may be interleaved on the
command line in any order with -C commands.

-C cmd Run the specified command only on the current open file. Multiple -C ar?
guments may be given and may be interleaved on the command line in any
order with -c commands.

-p prog Set the program name for prompts and some error messages, the default
value is xfs_io.

-f Create file if it does not already exist.

-r Open file read-only, initially. This is required if file is immutable or
append-only.

-i Start an idle thread. The purpose of this idle thread is to test io from
a multi threaded process. With single threaded process, the file table is
not shared and file structs are not reference counted. Spawning an idle
thread can help detecting file struct reference leaks.

-x Expert mode. Dangerous commands are only available in this mode. These
commands also tend to require additional privileges.

-V Prints the version number and exits.

The other open(2) options described below are also available from the command line.

CONCEPTS

xfs_io maintains a number of open files and memory mappings. Files can be ini?
tially opened on the command line (optionally), and additional files can also be
opened later.

xfs_io commands can be broken up into three groups. Some commands are aimed at do?
ing regular file I/O - read, write, sync, space preallocation, etc.

The second set of commands exist for manipulating memory mapped regions of a file -
mapping, accessing, storing, unmapping, flushing, etc.

The remaining commands are for the navigation and display of data structures relat?
ing to the open files, mappings, and the filesystems where they reside.

Many commands have extensive online help. Use the help command for more details on
any command.

FILE I/O COMMANDS

file [ N ]

Display a list of all open files and (optionally) switch to an alternate current open file.

open [[ -acdfrstRTPL ] path ]

Closes the current file, and opens the file specified by path instead. With? out any arguments, displays statistics about the current file - see the stat command.

-a opens append-only (O_APPEND).

-d opens for direct I/O (O_DIRECT).

-f creates the file if it doesn't already exist (O_CREAT).

-r opens read-only (O_RDONLY).

-s opens for synchronous I/O (O_SYNC).

-t truncates on open (O_TRUNC).

-n opens in non-blocking mode if possible (O_NONBLOCK).

-T create a temporary file not linked into the filesystem namespace (O_TMPFILE). The pathname passed must refer to a directory which is treated as virtual parent for the newly created invisible file. Can not be used together with the -r option.

-R marks the file as a realtime XFS file after opening it, if it is not already marked as such.

-P opens the path as a referent only (O_PATH). This is incompatible with other flags specifying other O_xxx flags apart from -L.

-L doesn't follow symlinks (O_NOFOLLOW). This is incompatible with other flags specifying other O_xxx flags apart from -P.

o See the open command.

close Closes the current open file, marking the next open file as current (if one exists).

c See the close command.

chmod -r | -w

Change the mode of the currently open file. The -r option will set the file permissions to read-only (0444), whilst the -w option will set the file per? missions to read-write (0644). This allows xfs_io to set up mismatches be?

tween the file permissions and the open file descriptor read/write mode to exercise permission checks inside various syscalls.

pread [ -b bsize ] [ -v ] [ -FBR [ -Z seed ] ] [ -V vectors ] offset length

Reads a range of bytes in a specified blocksize from the given offset.

-b can be used to set the blocksize into which the read(2) requests will be split. The default blocksize is 4096 bytes.

-v dump the contents of the buffer after reading, by default only the count of bytes actually read is dumped.

-F read the buffers in a forwards sequential direction.

-B read the buffers in a reserve sequential direction.

-R read the buffers in the give range in a random order.

-Z seed

specify the random number seed used for random reads.

-V vectors

Use the vectored IO read syscall preadv(2) with a number of blocksize length iovecs. The number of iovecs is set by the vectors parameter.

r See the pread command.

pwrite [ -i file ] [ -dDwNOW ] [ -s skip ] [ -b size ] [ -S seed ] [ -FBR [ -Z zeed ] ] [ -V vectors ] offset length

Writes a range of bytes in a specified blocksize from the given offset. The bytes written can be either a set pattern or read in from another file be? fore writing.

-i allows an input file to be specified as the source of the data to be written.

-d causes direct I/O, rather than the usual buffered I/O, to be used when reading the input file.

-w call fdatasync(2) once all writes are complete (included in timing results)

-N Perform the pwritev2(2) call with RWF_NOWAIT.

-D Perform the pwritev2(2) call with RWF_DSYNC.

-O perform pwrite once and return the (maybe partial) bytes written.

-W call fsync(2) once all writes are complete (included in timing re? sults)

-s specifies the number of bytes to skip from the start of the input file before starting to read.

-b used to set the blocksize into which the write(2) requests will be split. The default blocksize is 4096 bytes.

-S used to set the (repeated) fill pattern which is used when the data to write is not coming from a file. The default buffer fill pattern value is 0xcdcdcdcd.

-F write the buffers in a forwards sequential direction.

-B write the buffers in a reserve sequential direction.

-R write the buffers in the give range in a random order.

-Z seed

specify the random number seed used for random write

-V vectors

Use the vectored IO write syscall pwritev(2) with a number of block? size length iovecs. The number of iovecs is set by the vectors param? eter.

w See the pwrite command.

bmap [ -adelpv ] [ -n nx ]

Prints the block mapping for the current open file. Refer to the xfs_bmap(8) manual page for complete documentation.

fiemap [ -alv ] [ -n nx ] [ offset [ len ]]

Prints the block mapping for the current open file using the fiemap ioctl. Options behave as described in the xfs_bmap(8) manual page. Optionally, this command also supports passing the start offset from where to begin the mapping and the length of that region. The kernel will return any full extents which intersect with the requested range, and the fiemap command will print them in their entirety. If the requested range starts or ends in a hole, fiemap will print the hole, truncated to the requested range.

extsize [ -R | -D ] [ value ]

Display and/or modify the preferred extent size used when allocating space for the currently open file. If the -R option is specified, a recursive de? scent is performed for all directory entries below the currently open file

(-D can be used to restrict the output to directories only). If the target file is a directory, then the inherited extent size is set for that direc? tory (new files created in that directory inherit that extent size). The value should be specified in bytes, or using one of the usual units suffixes (k, m, g, b, etc). The extent size is always reported in units of bytes.

cowextsize [ -R | -D ] [ value ]

Display and/or modify the preferred copy-on-write extent size used when al? locating space for the currently open file. If the -R option is specified, a recursive descent is performed for all directory entries below the currently open file (-D can be used to restrict the output to directories only). If the target file is a directory, then the inherited CoW extent size is set for that directory (new files created in that directory inherit that CoW ex? tent size). The value should be specified in bytes, or using one of the usual units suffixes (k, m, g, b, etc). The extent size is always reported in units of bytes.

allocsp size 0

Sets the size of the file to size and zeroes any additional space allocated using the XFS_IOC_ALLOCSP/XFS_IOC_FREESP system call described in the xf? sctl(3) manual page. allocsp and freesp do exactly the same thing.

freesp size 0

See the allocsp command.

fadvise [ -r | -s | [[ -d | -n | -w ] offset length ]]

On platforms which support it, allows hints be given to the system regarding the expected I/O patterns on the file. The range arguments are required by some advise commands ([*] below), and the others must have no range argu? ments. With no arguments, the POSIX_FADV_NORMAL advice is implied (default readahead).

-d the data will not be accessed again in the near future (POSIX_FADV_DONTNEED[*]).

-n data will be accessed once and not be reused (POSIX_FADV_NOREUSE[*]).

-r expect access to data in random order (POSIX_FADV_RANDOM), which sets readahead to zero.

-s expect access to data in sequential order (POSIX_FADV_SEQUENTIAL),

which doubles the default readahead on the file.

-w advises the specified data will be needed again (POSIX_FADV_WILL? NEED[*]) which forces the maximum readahead.

fdatasync

Calls fdatasync(2) to flush the file's in-core data to disk.

fsync Calls fsync(2) to flush all in-core file state to disk.

s See the fsync command.

sync_range [ -a | -b | -w ] offset length

On platforms which support it, allows control of syncing a range of the file to disk. With no options, SYNC_FILE_RANGE_WRITE is implied on the range sup? plied.

-a wait for IO in the given range to finish after writing (SYNC_FILE_RANGE_WAIT_AFTER).

-b wait for IO in the given range to finish before writing (SYNC_FILE_RANGE_WAIT_BEFORE).

-w start writeback of dirty data in the given range (SYNC_FILE_RANGE_WRITE).

sync Calls sync(2) to flush all filesystems' in-core data to disk.

syncfs Calls syncfs(2) to flush this filesystem's in-core data to disk.

resvsp offset length

Allocates reserved, unwritten space for part of a file using the XFS_IOC_RESVSP system call described in the xfsctl(3) manual page.

unresvsp offset length

Frees reserved space for part of a file using the XFS_IOC_UNRESVSP system call described in the xfsctl(3) manual page.

falloc [ -k ] offset length

Allocates reserved, unwritten space for part of a file using the fallocate routine as described in the fallocate(2) manual page.

-k will set the FALLOC_FL_KEEP_SIZE flag as described in fallocate(2).

fcollapse offset length

Call fallocate with FALLOC_FL_COLLAPSE_RANGE flag as described in the fallo? cate(2) manual page to de-allocates blocks and eliminates the hole created in this process by shifting data blocks into the hole.

finsert offset length

Call fallocate with FALLOC_FL_INSERT_RANGE flag as described in the fallo?

cate(2) manual page to create the hole by shifting data blocks.

fpunch offset length

Punches (de-allocates) blocks in the file by calling fallocate with the FAL?

LOC_FL_PUNCH_HOLE flag as described in the fallocate(2) manual page.

funshare offset length

Call fallocate with FALLOC_FL_UNSHARE_RANGE flag as described in the fallo?

cate(2) manual page to unshare all shared blocks within the range.

fzero [ -k ] offset length

Call fallocate with FALLOC_FL_ZERO_RANGE flag as described in the fallo?

cate(2) manual page to allocate and zero blocks within the range. With the

-k option, use the FALLOC_FL_KEEP_SIZE flag as well.

zero offset length

Call xfsctl with XFS_IOC_ZERO_RANGE as described in the xfsctl(3) manual

page to allocate and zero blocks within the range.

truncate offset

Truncates the current file at the given offset using ftruncate(2).

sendfile -i srcfile | -f N [ offset length ]

On platforms which support it, allows a direct in-kernel copy between two

file descriptors. The current open file is the target, the source must be

specified as another open file (-f) or by path (-i).

readdir [ -v ] [ -o offset ] [ -l length ]

Read a range of directory entries from a given offset of a directory.

-v verbose mode - dump dirent content as defined in readdir(3)

-o specify starting offset

-l specify total length to read (in bytes)

seek -a | -d | -h [ -r ] [ -s ] offset

On platforms that support the lseek(2) SEEK_DATA and SEEK_HOLE options, dis?

play the offsets of the specified segments.

-a Display both data and hole segments starting at the specified offset.

-d Display the data segment starting at the specified offset.

-h Display the hole segment starting at the specified offset.

-r Recursively display all the specified segments starting at the speci?
fied offset.

-s Display the starting lseek(2) offset. This offset will be a calcu?
lated value when both data and holes are displayed together or per?
forming a recusively display.

reflink [ -C ] [ -q ] src_file [src_offset dst_offset length]

On filesystems that support the FICLONERANGE or BTRFS_IOC_CLONE_RANGE ioctls, map length bytes at offset dst_offset in the open file to the same physical blocks that are mapped at offset src_offset in the file src_file , replacing any contents that may already have been there. If a program writes into a reflinked block range of either file, the dirty blocks will be cloned, written to, and remapped ("copy on write") in the affected file, leaving the other file(s) unchanged. If src_offset, dst_offset, and length are omitted, all contents of src_file will be reflinked into the open file.

-C Print timing statistics in a condensed format.

-q Do not print timing statistics at all.

dedupe [ -C ] [ -q ] src_file src_offset dst_offset length

On filesystems that support the FIDEDUPERANGE or BTRFS_IOC_FILE_EXTENT_SAME ioctls, map length bytes at offset dst_offset in the open file to the same physical blocks that are mapped at offset src_offset in the file src_file , but only if the contents of both ranges are identical. This is known as block-based deduplication. If a program writes into a reflinked block range of either file, the dirty blocks will be cloned, written to, and remapped ("copy on write") in the affected file, leaving the other file(s) unchanged.

-C Print timing statistics in a condensed format.

-q Do not print timing statistics at all.

copy_range [ -s src_offset ] [ -d dst_offset ] [ -l length ] src_file | -f N

On filesystems that support the copy_file_range(2) system call, copies data from the source file into the current open file. The source must be speci? fied either by path (src_file) or as another open file (-f). If length is not specified, this command copies data from src_offset to the end of src_file into the dst_file at dst_offset.

-s Copy data from src_file beginning from src_offset.

-d Copy data into the open file beginning at dst_offset.

-l Copy up to length bytes of data.

swapext donor_file

Swaps extent forks between files. The current open file is the target. The donor file is specified by path. Note that file data is not copied (file content moves with the fork(s)).

set_encpolicy [ -c mode ] [ -n mode ] [ -f flags ] [ -v version ] [ keyspec ]

On filesystems that support encryption, assign an encryption policy to the current file. keyspec is a hex string which specifies the encryption key to use. For v1 encryption policies, keyspec must be a 16-character hex string (8 bytes). For v2 policies, keyspec must be a 32-character hex string (16 bytes). If unspecified, an all-zeroes value is used.

-c mode

contents encryption mode (e.g. AES-256-XTS)

-n mode

filenames encryption mode (e.g. AES-256-CTS)

-f flags

policy flags (numeric)

-v version

policy version. Defaults to 1 or 2 depending on the length of keyspec; or to 1 if keyspec is unspecified.

get_encpolicy [ -1 ] [ -t ]

On filesystems that support encryption, display the encryption policy of the current file.

-1 Use only the old ioctl to get the encryption policy. This only works if the file has a v1 encryption policy.

-t Test whether v2 encryption policies are supported. Prints "supported", "unsupported", or an error message.

add_enckey [ -d descriptor ]

On filesystems that support encryption, add an encryption key to the filesystem containing the currently open file. The key in binary (typically 64 bytes long) is read from standard input.

-d descriptor

key descriptor, as a 16-character hex string (8 bytes). If given, the key will be available for use by v1 encryption policies that use this descriptor. Otherwise, the key is added as a v2 policy key, and on success the resulting "key identifier" will be printed.

rm_enckey [ -a ] keyspec

On filesystems that support encryption, remove an encryption key from the filesystem containing the currently open file. keyspec is a hex string specifying the key to remove, as a 16-character "key descriptor" or a 32-character "key identifier".

-a Remove the key for all users who have added it, not just the current user. This is a privileged operation.

enckey_status keyspec

On filesystems that support encryption, display the status of an encryption key. keyspec is a hex string specifying the key for which to display the status, as a 16-character "key descriptor" or a 32-character "key identi? fier".

lsattr [ -R | -D | -a | -v ]

List extended inode flags on the currently open file. If the -R option is specified, a recursive descent is performed for all directory entries below the currently open file (-D can be used to restrict the output to directo? ries only). This is a depth first descent, it does not follow symlinks and it also does not cross mount points.

chattr [ -R | -D ] [ +/-riasAdtPneEfSxC ]

Change extended inode flags on the currently open file. The -R and -D op? tions have the same meaning as above. The mapping between each letter and the inode flags (refer to xfsctl(3) for the full list) is available via the help command.

flink path

Link the currently open file descriptor into the filesystem namespace.

stat [ -v|-r ]

Selected statistics from stat(2) and the XFS_IOC_GETXATTR system call on the current file. If the -v option is specified, the atime (last access), mtime (last modify), and ctime (last change) timestamps are also displayed. The

-r option dumps raw fields from the stat structure.

statx [ -v|-r ][ -m basic | -m all | -m <mask> ][ -FD ]

Selected statistics from stat(2) and the XFS_IOC_GETXATTR system call on the current file.

-v Show timestamps.

-r Dump raw statx structure values.

-m basic

Set the field mask for the statx call to STATX_BASIC_STATS.

-m all

Set the the field mask for the statx call to STATX_ALL (default).

-m <mask>

Specify a numeric field mask for the statx call.

-F Force the attributes to be synced with the server.

-D Don't sync attributes with the server.

chproj [ -R|-D ]

Modifies the project identifier associated with the current path. The -R option will recursively descend if the current path is a directory. The -D option will also recursively descend, only setting modifying projects on subdirectories. See the xfs_quota(8) manual page for more information about project identifiers.

lsproj [ -R|-D ]

Displays the project identifier associated with the current path. The -R and -D options behave as described above, in chproj.

parent [ -cpv ]

By default this command prints out the parent inode numbers, inode generation numbers and basenames of all the hardlinks which point to the inode of the current file.

-p the output is similar to the default output except pathnames up to the mount-point are printed out instead of the component name.

-c the file's filesystem will check all the parent attributes for consistency.

-v verbose output will be printed.

[NOTE: Not currently operational on Linux.]

utimes atime_sec atime_nsec mtime_sec mtime_nsec

The utimes command changes the atime and mtime of the current file. sec uses UNIX timestamp notation and is the seconds elapsed since 1970-01-01 00:00:00 UTC. nsec is the nanoseconds since the sec. This value needs to be in the range 0-999999999 with UTIME_NOW and UTIME_OMIT being exceptions. Each (sec, nsec) pair constitutes a single timestamp value.

MEMORY MAPPED I/O COMMANDS

mmap [ N | [[ -rwxS ] [-s size ] offset length ]]

With no arguments, mmap shows the current mappings. Specifying a single nu? meric argument N sets the current mapping. If two arguments are specified (a range specified by offset and length), a new mapping is created spanning the range, and the protection mode can be given as a combination of PROT_READ (-r), PROT_WRITE (-w), and PROT_EXEC (-x). The mapping will be created with the MAP_SHARED flag by default, or with the Linux specific (MAP_SYNC | MAP_SHARED_VALIDATE) flags if -S is given. -s size is used to do a mmap(size) && munmap(size) operation at first, try to reserve some ex? tendible free memory space, if size is bigger than length parameter. But there's not guarantee that the memory after length ( up to size ) will stay free. e.g. "mmap -rw -s 8192 1024" will mmap 0 ~ 1024 bytes memory, but try to reserve 1024 ~ 8192 free space(no guarantee). This free space will helpful for "mremap 8192" without MREMAP_MAYMOVE flag.

mm See the mmap command.

mremap [ -f <new_address> ] [ -m ] new_length

Changes the current mapping size to new_length. Whether the mapping may be moved is controlled by the flags passed; MREMAP_FIXED (-f), or MREMAP_MAY? MOVE (-m). new_length specifies a page-aligned address to which the mapping must be moved. It can be set to 139946004389888, 4096k or 1g etc.

mrm See the mremap command.

munmap Unmaps the current memory mapping.

mu See the munmap command.

mread [ -f | -v ] [ -r ] [ offset length ]

Accesses a segment of the current memory mapping, optionally dumping it to the standard output stream (with -v or -f option) for inspection. The ac?

cesses are performed sequentially from the start offset by default, but can also be done from the end backwards through the mapping if the -r option in specified. The two verbose modes differ only in the relative offsets they display, the -f option is relative to file start, whereas -v shows offsets relative to the start of the mapping.

mr See the mread command.

mwrite [ -r ] [ -S seed ] [ offset length ]

Stores a byte into memory for a range within a mapping. The default stored value is 'X', repeated to fill the range specified, but this can be changed using the -S option. The memory stores are performed sequentially from the start offset by default, but can also be done from the end backwards through the mapping if the -r option in specified.

mw See the mwrite command.

msync [ -i ] [ -a | -s ] [ offset length ]

Writes all modified copies of pages over the specified range (or entire map? ping if no range specified) to their backing storage locations. Also, op? tionally invalidates (-i) so that subsequent references to the pages will be obtained from their backing storage locations (instead of cached copies). The flush can be done synchronously (-s) or asynchronously (-a).

ms See the msync command.

madvise [ -d | -r | -s | -w ] [ offset length ]

Modifies page cache behavior when operating on the current mapping. The range arguments are required by some advise commands ([*] below). With no arguments, the POSIX_MADV_NORMAL advice is implied (default readahead).

-d the pages will not be needed (POSIX_MADV_DONTNEED[*]).

-r expect random page references (POSIX_MADV_RANDOM), which sets reada? head to zero.

-s expect sequential page references (POSIX_MADV_SEQUENTIAL), which dou? bles the default readahead on the file.

-w advises the specified pages will be needed again (POSIX_MADV_WILL? NEED[*]) which forces the maximum readahead.

mincore

Dumps a list of pages or ranges of pages that are currently in core, for the

current memory mapping.

FILESYSTEM COMMANDS

bulkstat [ -a agno ] [ -d ] [ -e endino ] [ -n batchsize ] [ -s startino ] [ -v version"]

Display raw stat information about a bunch of inodes in an XFS filesystem.

Options are as follows:

-a agno

Display only results from the given allocation group. If not specified, all results returned will be displayed.

-d Print debugging information about call results.

-e endino

Stop displaying records when this inode number is reached. De? faults to stopping when the system call stops returning results.

-n batchsize

Retrieve at most this many records per call. Defaults to 4,096.

-s startino

Display inode allocation records starting with this inode. De? faults to the first inode in the filesystem. If the given inode is not allocated, results will begin with the next allocated inode in the filesystem.

-v version

Use a particular version of the kernel interface. Currently sup? ported versions are 1 and 5.

bulkstat_single [ -d ] [ -v version ] [ inum... | special... ]

Display raw stat information about individual inodes in an XFS filesystem. The -d and -v options are the same as the bulkstat command. Arguments must be inode numbers or any of the special values:

root Display information about the root directory inode.

freeze Suspend all write I/O requests to the filesystem of the current file. Only available in expert mode and requires privileges.

thaw Undo the effects of a filesystem freeze operation. Only available in expert mode and requires privileges.

inject [ tag ]

Inject errors into a filesystem to observe filesystem behavior at specific points under adverse conditions. Without the tag argument, displays the list of error tags available. Only available in expert mode and requires privi? leges.

resblks [ blocks ]

Get and/or set count of reserved filesystem blocks using the XFS_IOC_GET_RESBLKS or XFS_IOC_SET_RESBLKS system calls. Note -- this can be useful for exercising out of space behavior. Only available in expert mode and requires privileges.

shutdown [ -f ]

Force the filesystem to shut down, preventing any further IO. XFS and other filesystems implement this functionality, although implementation details may differ slightly. Only available in expert mode and requires privileges. By default, the filesystem will not attempt to flush completed transactions to disk before shutting down the filesystem. This simulates a disk failure or crash.

-f Force the filesystem to flush all completed transactions to disk be? fore shutting down, matching XFS behavior when critical corruption is encountered.

statfs Selected statistics from statfs(2) and the XFS_IOC_FSGEOMETRY system call on the filesystem where the current file resides.

inode [ [ -n ] number ] [ -v ]

The inode command queries physical information about an inode. With no argu? ments, it will return 1 or 0, indicating whether or not any inode numbers greater than 32 bits are currently in use in the filesystem. If given an inode number as an argument, the command will return the same inode number if it is in use, or 0 if not. With -n number , the next used inode number after this number will be returned, or zero if the supplied inode number is the highest one in use. With -v the command will also report the number of bits (32 or 64) used by the inode number printed in the result; if no inode number was specified on the command line, the maximum possible inode number in the system will be printed along with its size.

inumbers [ -a agno ] [ -d ] [ -e endino ] [ -n batchsize ] [ -s startino ] [ -v

version ]

Prints allocation information about groups of inodes in an XFS filesystem. Callers can use this information to figure out which inodes are allocated. Options are as follows:

-a agno

Display only results from the given allocation group. If not specified, all results returned will be displayed.

-d Print debugging information about call results.

-e endino

Stop displaying records when this inode number is reached. De? faults to stopping when the system call stops returning results.

-n batchsize

Retrieve at most this many records per call. Defaults to 4,096.

-s startino

Display inode allocation records starting with this inode. De? faults to the first inode in the filesystem. If the given inode is not allocated, results will begin with the next allocated inode in the filesystem.

-v version

Use a particular version of the kernel interface. Currently sup? ported versions are 1 and 5.

scrub type [ agnumber | ino gen ]

Scrub internal XFS filesystem metadata. The type parameter specifies which type of metadata to scrub. For AG metadata, one AG number must be speci? fied. For file metadata, the scrub is applied to the open file unless the inode number and generation number are specified.

repair type [ agnumber | ino gen ]

Repair internal XFS filesystem metadata. The type parameter specifies which type of metadata to repair. For AG metadata, one AG number must be speci? fied. For file metadata, the repair is applied to the open file unless the inode number and generation number are specified.

label [ -c | -s label ]

On filesystems that support online label manipulation, get, set, or clear

the filesystem label. With no options, print the current filesystem label.

The -c option clears the filesystem label by setting it to the null string.

The -s label option sets the filesystem label to label. If the label is longer than the filesystem will accept, xfs_io will print an error message.

XFS filesystem labels can be at most 12 characters long.

fsmap [ -d | -l | -r ] [ -m | -v ] [ -n nx ] [ start ] [ end ]

Prints the mapping of disk blocks used by the filesystem hosting the current file. The map lists each extent used by files, allocation group metadata, journalling logs, and static filesystem metadata, as well as any regions that are unused. Each line of the listings takes the following form:

extent: major:minor [startblock..endblock]: owner startoffset..endoffset length

Static filesystem metadata, allocation group metadata, btrees, journalling logs, and free space are marked by replacing the startoffset..endoffset with the appropriate marker. All blocks, offsets, and lengths are specified in units of 512-byte blocks, no matter what the filesystem's block size is.

The optional start and end arguments can be used to constrain the output to a particular range of disk blocks. If these two options are specified, ex? actly one of -d, -l, or -r must also be set.

-d Display only extents from the data device. This option only ap? plies for XFS filesystems.

-l Display only extents from the external log device. This option only applies to XFS filesystems.

-r Display only extents from the realtime device. This option only applies to XFS filesystems.

-m Display results in a machine readable format (CSV). This option is not compatible with the -v flag. The columns of the output are: extent number, device major, device minor, physical start, physical end, owner, offset start, offset end, length. The start, end, and length numbers are provided in units of 512b. The owner field is a special string that takes the form:

inode_%lld_data

for inode data.

inode_%lld_data_bmbt

for inode data extent maps.

inode_%lld_attr

for inode extended attribute data.

inode_%lld_attr_bmbt

for inode extended attribute extent maps.

special_%u:%u

for other filesystem metadata.

-n num_extents

If this option is given, fsmap obtains the extent list of the file in groups of num_extents extents. In the absence of -n, fsmap queries the system for extents in groups of 131,072 records.

-v Shows verbose information. When this flag is specified, addi? tional AG specific information is appended to each line in the following form:

agno (startagblock..endagblock) nblocks flags

A second -v option will print out the flags legend. This option is not compatible with the -m flag.

OTHER COMMANDS

help [ command ]

Display a brief description of one or all commands.

print Display a list of all open files and memory mapped regions. The current file and current mapping are distinguishable from any others.

p See the print command.

quit Exit xfs_io.

q See the quit command.

log_writes -d device -m mark

Create a mark named mark in the dm-log-writes log specified by device. This is intended to be equivalent to the shell command:

dmsetup message device 0 mark mark

lw See the log_writes command.

crc32cselftest

Test the internal crc32c implementation to make sure that it computes re?

sults correctly.

SEE ALSO

mkfs.xfs(8), xfsctl(3), xfs_bmap(8), xfs_db(8), xfs(5), fdatasync(2), fstat(2), fs?tatfs(2), fsync(2), ftruncate(2), futimens(3), mmap(2), msync(2), open(2), pread(2), pwrite(2), readdir(3), dmsetup(8).