NAME

debugfs - ext2/ext3/ext4 file system debugger

SYNOPSIS

debugfs [**-DVwci**] [**-b** blocksize] [**-s** superblock] [**-f** cmd_file] [**-R** request] [**-d** data_source_device] [device]

DESCRIPTION

The **debugfs** program is an interactive file system debugger. It can be used to examine and change the state of an ext2, ext3, or ext4 file system.

device is the special file corresponding to the device containing the file system (e.g /dev/hdXX).

OPTIONS

-w Specifies that the file system should be opened in read-write mode. Without this option, the file system is opened in read-only mode.

-c Specifies that the file system should be opened in catastrophic mode, in which the inode and group bitmaps are not read initially. This can be useful for filesystems with significant corruption, but because of this, catastrophic mode forces the filesystem to be opened read-only.

-i Specifies that *device* represents an ext2 image file created by the **e2image** program. Since the ext2 image file only contains the superblock, block group descriptor, block and inode allocation bitmaps, and the inode table, many **debugfs** commands will not function properly. **Warning:** no safety checks are in place, and **debugfs** may fail in interesting ways if commands such as *ls*, *dump*, etc. are tried without specifying the *data_source_device* using the *-d* option. **debugfs** is a debugging tool. It has rough edges!

-d data_source_device
Used with the *-i* option, specifies that *data_source_device* should be used when reading blocks not found in the ext2 image file. This includes data, directory, and indirect blocks.

-b blocksize
Forces the use of the given block size for the file system, rather than detecting the correct block size as normal.

-s superblock
Causes the file system superblock to be read from the given block number, instead of using the primary superblock (located at an offset of 1024 bytes from the beginning of the filesystem). If you specify the *-s* option, you must also provide the blocksize of the filesystem via the *-b* option.

-f cmd_file
Causes **debugfs** to read in commands from *cmd_file*, and execute them. When **debugfs** is finished executing those commands, it will exit.

-D Causes **debugfs** to open the device using Direct I/O, bypassing the buffer cache. Note that some Linux devices, notably device mapper as of this writing, do not support Direct I/O.

-R request
Causes **debugfs** to execute the single command *request*, and then exit.

-V print the version number of **debugfs** and exit.

SPECIFYING FILES

Many **debugfs** commands take a *filespec* as an argument to specify an inode (as opposed to a pathname) in the filesystem which is currently opened by **debugfs**. The*filesp ec* argument may be specified in two forms. The first form is an inode number surrounded by angle brackets, e.g., <2>. The second form is a pathname; if the pathname is prefixed by a forward slash ('/'), then it is interpreted relative to the root of the filesystem which is currently opened by **debugfs**. If not, the pathname is interpreted relative to the current working directory as maintained by **debugfs**.

This may be modified by using the **debugfs** command *cd*.

COMMANDS

This is a list of the commands which **debugfs** supports.

blocks *filespace*
Print the blocks used by the inode *filespec* to stdout.

bmap *filespec logical_block*
Print the physical block number corresponding to the logical block number *logical_block* in the inode *filespec*.

block_dump *[-f filespec] block_num*
Dump the filesystem block given by *block_num* in hex and ASCII format to the console. If the *-f* option is specified, the block number is relative to the start of the given **filespec**.

cat *filespec*
Dump the contents of the inode *filespec* to stdout.

cd *filespec*
Change the current working directory to *filespec*.

chroot *filespec*
Change the root directory to be the directory *filespec*.

close *[-a]*
Close the currently open file system. If the *-a* option is specified, write out any changes to the superblock and block group descriptors to all of the backup superblocks, not just to the master superblock.

clri *filespec*
Clear the contents of the inode *filespec*.

dirsearch *filespec filename*
Search the directory *filespec* for *filename*.

dirty Mark the filesystem as dirty, so that the superblocks will be written on exit.

dump *[-p] filespec out_file*
Dump the contents of the inode *filespec* to the output file *out_file*. If the*-p* option is given set the owner, group and permissions information on *out_file* to match *filespec*.

dump_mmp
Display the multiple-mount protection (mmp) field values.

dx_hash *[-h hash_alg] [-s hash_seed] filename*
Calculate the directory hash of *filename*. The hash algorithm specified with*-h* ma y be **legacy**, **half_md4**, or **tea**. The hash seed specified with*-s* m ust be in UUID format.

dump_extents *[-n] [-l] filespec*
Dump the the extent tree of the inode *filespec*. The*-n* flag will cause **dump_exten ts** to only display the interior nodes in the extent tree. The *-l* flag will cause **dump_extents** to only display the leaf nodes in the extent tree.

(Please note that the length and range of blocks for the last extent in an interior node is an estimate by the extents library functions, and is not stored in filesystem data structures. Hence, the values displayed may not necessarily by accurate and does not indicate a problem or corruption in the file system.)

expand_dir *filespec*
Expand the directory *filespec*.

feature *[fs_feature] [-fs_feature] ...*
Set or clear various filesystem features in the superblock. After setting or clearing any filesystem features that were requested, print the current state of the filesystem feature

set.

filefrag *[-dvr] filespec*
Print the number of contiguous extents in *filespec*. If*filesp ec* is a directory and the *-d* option is not specified, *filefrag* will print the number of contiguous extents for each file in the directory. The *-v* option will cause *filefrag* print a tabular listing of the contiguous extents in the file. The *-r* option will cause *filefrag* to do a recursive listing of the directory.

find_free_block *[count [goal]]*
Find the first *count* free blocks, starting from *goal* and allocate it. Also available as **ffb**.

find_free_inode *[dir [mode]]*
Find a free inode and allocate it. If present, *dir* specifies the inode number of the directory which the inode is to be located. The second optional argument *mode* specifies the permissions of the new inode. (If the directory bit is set on the mode, the allocation routine will function differently.) Also available as **ffi**.

freeb *block [count]*
Mark the block number *block* as not allocated. If the optional argument *count* is present, then *count* blocks starting at block number *block* will be marked as not allocated.

freefrag *[-c chunk_kb]*
Report free space fragmentation on the currently open file system. If the *-c* option is specified then the filefrag command will print how many free chunks of size *chunk_kb* can be found in the file system. The chunk size must be a power of two and be larger than the file system block size.

freei *filespec [num]*
Free the inode specified by *filespec*. If*num* is specified, also clear n um-1 inodes after the specified inode.

help Print a list of commands understood by **debugfs**.

htree_dump *filespec*
Dump the hash-indexed directory *filespec*, showing its tree structure.

icheck *block ...*
Print a listing of the inodes which use the one or more blocks specified on the command line.

inode_dump *filespec*
Print the contents of the inode data structure in hex and ASCII format.

imap *filespec*
Print the location of the inode data structure (in the inode table) of the inode *filespec*.

init_filesys *device blocksize*
Create an ext2 file system on *device* with device size *blocksize*. Note that this does not fully initialize all of the data structures; to do this, use the mke2fs(8) program. This is just a call to the low-level library, which sets up the superblock and block descriptors.

kill_file *filespec*
Deallocate the inode *filespec* and its blocks. Note that this does not remove any directory entries (if any) to this inode. See the rm(1) command if you wish to unlink a file.

lcd *directory*
Change the current working directory of the **debugfs** process to *directory* on the native filesystem.

ln *filespec dest_file*
Create a link named *dest_file* which is a hard link to *filespec*. Note this does not adjust the inode reference counts.

logdump *[-acs] [-b block] [-i filespec] [-f journal_file] [output_file]*
Dump the contents of the ext3 journal. By default, dump the journal inode as specified in the superblock. However, this can be overridden with the *-i* option, which dumps the journal from the internal inode given by *filespec*. A regular file containing journal data can be specified using the *-f* option. Finally, the *-s* option utilizes the backup information in the superblock to locate the journal.

The *-a* option causes the **logdump** program to print the contents of all of the descriptor blocks. The*-b* option causes **logdump** to prin t all journal records that are refer to the specified block. The*-c* option will prin t out the contents of all of the data blocks selected by the *-a* and *-b* options.

ls *[-d] [-l] [-p] filespec*
Print a listing of the files in the directory *filespec*. The*-d* flag will list deleted en tries in the directory. The*-l* flag will list files using a more v erbose format. The *-p* flag will list the files in a format which is more easily parsable by scripts, as well as making it more clear when there are spaces or other non-printing characters at the end of filenames.

list_deleted_inodes *[limit]*
List deleted inodes, optionally limited to those deleted within *limit* seconds ago. Also available as **lsdel**.

This command was useful for recovering from accidental file deletions for ext2 file systems. Unfortunately, it is not useful for this purpose if the files were deleted using ext3 or ext4, since the inode's data blocks are no longer available after the inode is released.

modify_inode *filespec*
Modify the contents of the inode structure in the inode *filespec*. Also available as **mi**.

mkdir *filespec*
Make a directory.

mknod *filespec [p|[c|b] major minor]]*
Create a special device file (a named pipe, character or block device). If a character or block device is to be made, the *major* and *minor* device numbers must be specified.

ncheck *[-c] inode_num ...*
Take the requested list of inode numbers, and print a listing of pathnames to those inodes. The *-c* flag will enable checking the file type information in the directory entry to make sure it matches the inode's type.

open *[-weficD] [-b blocksize] [-s superblock] device*
Open a filesystem for editing. The *-f* flag forces the filesystem to be opened even if there are some unknown or incompatible filesystem features which would normally prevent the filesystem from being opened. The *-e* flag causes the filesystem to be opened in exclusive mode. The *-b*, *-c*, *-i*, *-s*, *-w*, and *-D* options behave the same as the command-line options to **debugfs**.

punch *filespec start_blk [end_blk]*
Delete the blocks in the inode ranging from *start_blk* to *end_blk*. If*end_blk* is omitted then this command will function as a truncate command; that is, all of the blocks starting at *start_blk* through to the end of the file will be deallocated.

symlink *filespec target*
Make a symbolic link.

pwd Print the current working directory.

quit Quit **debugfs**

rdump *directory[...] destination*
Recursively dump *directory*, or multiple *directories*, and all its contents (including regular files, symbolic links, and other directories) into the named *destination*, which should be

an existing directory on the native filesystem.

rm *pathname*
Unlink *pathname*. If this causes the inode pointed to by *pathname* to have no other references, deallocate the file. This command functions as the unlink() system call.

rmdir *filespec*
Remove the directory *filespec*.

setb *block [count]*
Mark the block number *block* as allocated. If the optional argument *count* is present, then *count* blocks starting at block number *block* will be marked as allocated.

set_block_group *bgnum field value*
Modify the block group descriptor specified by *bgnum* so that the block group descriptor field *field* has value *value*. Also available as **set_bg**.

seti *filespec [num]*
Mark inode *filespec* as in use in the inode bitmap. If *num* is specified, also set num-1 inodes after the specified inode.

set_inode_field *filespec field value*
Modify the inode specified by *filespec* so that the inode field *field* has value *value*. The list of valid inode fields which can be set via this command can be displayed by using the command: **set_inode_field -l** Also available as **sif**.

set_mmp_value *field value*
Modify the multiple-mount protection (MMP) data so that the MMP field *field* has value *value*. The list of valid MMP fields which can be set via this command can be displayed by using the command: **set_mmp_value -l** Also available as **smmp**.

set_super_value *field value*
Set the superblock field *field* to *value*. The list of valid superblock fields which can be set via this command can be displayed by using the command: **set_super_value -l** Also available as **ssv**.

show_super_stats *[-h]*
List the contents of the super block and the block group descriptors. If the *-h* flag is given, only print out the superblock contents. Also available as **stats**.

stat *filespec*
Display the contents of the inode structure of the inode *filespec*.

testb *block [count]*
Test if the block number *block* is marked as allocated in the block bitmap. If the optional argument *count* is present, then *count* blocks starting at block number *block* will be tested.

testi *filespec*
Test if the inode *filespec* is marked as allocated in the inode bitmap.

undel <*inode_number*> *[pathname]*
Undelete the specified inode number (which must be surrounded by angle brackets) so that it and its blocks are marked in use, and optionally link the recovered inode to the specified pathname. The **e2fsck** command should always be run after using the **undel** command to recover deleted files.

Note that if you are recovering a large number of deleted files, linking the inode to a directory may require the directory to be expanded, which could allocate a block that had been used by one of the yet-to-be-undeleted files. So it is safer to undelete all of the inodes without specifying a destination pathname, and then in a separate pass, use the debugfs **link** command to link the inode to the destination pathname, or use **e2fsck** to check the filesystem and link all of the recovered inodes to the lost+found directory.

unlink *pathname*
Remove the link specified by *pathname* to an inode. Note this does not adjust the inode reference counts.

write *source_file out_file*
Copy the contents of *source_file* into a newly-created file in the filesystem named *out_file*.

zap_block *[-f filespec] [-o offset] [-l length] [-p pattern] block_num*

Overwrite the block specified by
block_num with zero (NUL) bytes, or if *-p* is given use the byte specified by *pattern*. If *-f* is given then *block_num* is relative to the start of the file given by *filespec*. The *-o* and *-l* options limit the range of bytes to zap to the specified *offset* and *length* relative to the start of the block.

zap_block *[-f filespec] [-b bit] block_num*
Bit-flip portions of the physical *block_num*. If *-f* is giv en, then *block_num* is a logical block relative to the start of *filespec*.

ENVIRONMENT VARIABLES

DEBUGFS_PAGER, PAGER

The **debugfs** program always pipes the output of the some commands through a pager program. These commands include: *show_super_stats* (*stats*), *list_directory* (*ls*), *show_inode_info* (*stat*), *list_deleted_inodes* (*lsdel*), and *htree_dump*. The specific pager can explicitly specified by the **DEBUGFS_PAGER** environment variable, and if it is not set, by the **PAGER** environment variable.

Note that since a pager is always used, the less(1) pager is not particularly appropriate, since it clears the screen before displaying the output of the command and clears the output the screen when the pager is exited. Many users prefer to use the less(1) pager for most purposes, which is why the **DEBUGFS_PAGER** environment variable is available to override the more general **PAGER** environment variable.

AUTHOR

debugfs was written by Theodore Ts'o <tytso@mit.edu>.

SEE ALSO

dumpe2fs(8), tune2fs(8), e2fsck(8), mke2fs(8), ext4(5)