Appendix A

Repairing EFS Filesystem ProblemsWith *fsck*

The *fsck* command checks EFS filesystem consistency and sometimes repairs problems that are found. It is not used on XFS filesystems. This appendix describes the messages that are produced by each phase of *fsck*, what they mean, and what you should do about each one.

The sections in this appendix are:

- "Initialization Phase" on page 200
- "General Errors" on page 201
- "Phase 1 Check Blocks and Sizes" on page 201
- "Phase 2 Check Pathnames" on page 205
- "Phase 3 Check Connectivity" on page 207
- "Phase 4 Check Reference Counts" on page 209
- "Phase 5 Check Free List" on page 213
- "Phase 6 Salvage Free List" on page 214
- "Cleanup Phase" on page 214

The following abbreviations are used in *fsck* error messages:

BLK	block number
DUP	duplicate block number
DIR	directory name
MTIME	time file was last modified
UNREF	unreferenced

The following sections use these single-letter abbreviations:

B	block number
F	file (or directory) name
I	inode number
M	file mode
O	user ID of a file's owner
S	file size
T	time file was last modified
X	link count, or number of BAD, DUP, or MISSING blocks, or number of files (depending on context)
Y	corrected link count number, or number of blocks in filesystem (depending on context)
Z	number of free blocks

In actual *fsck* output, these abbreviations are replaced by the appropriate numbers.

Initialization Phase

The command line syntax is checked. Before the filesystem check can be performed, *fsck* sets up some tables and opens some files. The *fsck* command terminates if there are initialization errors.

General Errors

Two error messages may appear in any phase. Although *fsck* prompts for you to continue checking the filesystem, it is generally best to regard these errors as fatal. Stop the command and investigate what may have caused the problem.

`CAN NOT READ: BLK` *B* `(CONTINUE?)`
: The request to read a specified block number *B* in the filesystem failed. This error indicates a serious problem, probably a hardware failure or an error that causes *fsck* to try to read a block that is not in the filesystem. Press **n** to stop *fsck*. Shut down the system to the System Maintenance Menu and run hardware diagnostics on the disk drive and controller.

`CAN NOT WRITE: BLK` *B* `(CONTINUE?)`
: The request for writing a specified block number *B* in the filesystem failed. The disk may be write-protected or there may be a hardware problem. Press **n** to stop *fsck*. Check to make sure the disk is not set to "read only." (Some, though not all, disks have this feature.) If the disk is not write-protected, shut down the system to the System Maintenance Menu and run hardware diagnostics on the disk drive and controller.

Phase 1 Check Blocks and Sizes

This phase checks the inode list. It reports error conditions resulting from:

- checking inode types
- setting up the zero-link-count table
- examining inode block numbers for bad or duplicate blocks
- checking inode size
- checking inode format

Phase 1 Error Messages

Phase 1 has three types of error messages: information messages, messages with a CONTINUE? prompt, and messages with a CLEAR? prompt. The responses that you give to phase 1 prompts affect *fsck* functions. The possible responses are discussed in the next section, "Phase 1 Responses." Typically, the right answer is yes, except as noted.

`UNKNOWN FILE TYPE I=`*I* `(CLEAR?)`
: The mode word of the inode *I* suggests that the inode is not a pipe, special character inode, regular inode, directory inode, symbolic link, or socket.

`LINK COUNT TABLE OVERFLOW (CONTINUE?)`
: There is no more room in an internal table for *fsck* containing allocated inodes with a link count of zero.

B `BAD I=`*I*
: Inode *I* contains block number *B* with a number lower than the number of the first data block in the filesystem or greater than the number of the last block in the filesystem. This error condition may invoke the EXCESSIVE BAD BLKS error condition in Phase 1 if inode *I* has too many block numbers outside the filesystem range. This error condition invokes the BAD/DUP error condition in Phase 2 and Phase 4.

`EXCESSIVE BAD BLOCKS I=`*I* `(CONTINUE?)`
: There is more than a tolerable number (usually 50) of blocks with a number lower than the number of the first data block in the filesystem or greater than the number of the last block in the filesystem associated with inode *I*.

B `DUP I=`*I*
: Inode *I* contains block number *B*, which is already claimed by another inode. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if inode *I* has too many block numbers claimed by other inodes. This error condition invokes Phase 1B and the BAD/DUP error condition in Phase 2 and Phase 4. Typically, you should answer no the first time this error appears and yes the second time if you know the files claimed by the other inode.

`EXCESSIVE DUP BLKS I=`*I* `(CONTINUE?)`
: There is more than a tolerable number (usually 50) of blocks claimed by other inodes.

`DUP TABLE OVERFLOW (CONTINUE?)`
: There is no more room in an internal table in *fsck* containing duplicate block numbers.

`PARTIALLY ALLOCATED INODE I=`*I* `(CLEAR?)`
: Inode *I* is neither allocated nor unallocated.

`RIDICULOUS NUMBER OF EXTENTS (`*n*`) (max allowed `*n*`)`
: The number of extents is larger than the maximum the system can set and is therefore ridiculous.

`ILLEGAL NUMBER OF INDIRECT EXTENTS (n)`
: The number of extents or pointers to extents (indirect extents) exceeds the number of slots in the inode for describing extents.

`BAD MAGIC IN EXTENT`
: The pointer to an extent contains a "magic number." If this number is invalid, the pointer to the extent is probably corrupt.

`EXTENT OUT OF ORDER`
: An extent's idea of where it is in the file is inconsistent with the extent pointer in relation to other extent pointers.

`ZERO LENGTH EXTENT`
: An extent is zero length.

`ZERO SIZE DIRECTORY`
: It is erroneous for a directory inode to claim a size of zero. The corresponding inode is cleared.

`DIRECTORY SIZE ERROR`
: A directory's size must be an integer number of blocks. The size is recomputed based on its extents.

`DIRECTORY EXTENTS CORRUPTED`
: If the computation of size (above) fails, *fsck* prints this message and asks to clear the inode.

`NUMBER OF EXTENTS TOO LARGE`
: The number of extents or pointers to extents (indirect extents) exceeds the number of slots in the inode for describing extents.

`POSSIBLE DIRECTORY SIZE ERROR`
: The number of blocks in the directory computed from extent pointer lengths is inconsistent with the number computed from the inode size field.

`POSSIBLE FILE SIZE ERROR`
: The number of blocks in the file computed from extent pointer lengths is inconsistent with the number computed from the inode size field. *fsck* gives the option of clearing the inode in this case.

Phase 1 Responses

Table A-1 explains the significance of responses to Phase 1 prompts:

Table A-1 Meaning of *fsck* Phase 1 Responses

Prompt	Response	Meaning
CONTINUE?	n	Terminate the command.
CONTINUE?	y	Continue with the command. This error condition means that a complete check of the filesystem is not possible. A second run of *fsck* should be made to recheck this filesystem.
CLEAR?	n	Ignore the error condition. A “no” response is appropriate only if the user intends to take other measures to fix the problem.
CLEAR?	y	Deallocate inode *I* by zeroing its contents. This may invoke the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode.

Phase 1B Rescan for More Bad Dups

When a duplicate block is found in the filesystem, the filesystem is rescanned to find the inode that previously claimed that block. When the duplicate block is found, the following information message is printed:

B `DUP I=`*I* Inode *I* contains block number *B*, which is already claimed by another inode. This error condition invokes the BAD/DUP error condition in Phase 2. Inodes with overlapping blocks may be determined by examining this error condition and the DUP error condition in Phase 1.

Phase 2 Check Pathnames

This phase traverses the pathname tree, starting at the root directory. *fsck* examines each inode that is being used by a file in a directory of the filesystem being checked.

Referenced files are marked in order to detect unreferenced files later on. The command also accumulates a count of all links, which it checks against the link counts found in Phase 4.

Phase 2 reports error conditions resulting from the following:

- root inode mode and status incorrect
- directory inode pointers out of range
- directory entries pointing to bad inodes

fsck examines the root directory inode first, since this directory is where the search for all pathnames must start.

If the root directory inode is corrupted, or if its type is not *directory, fsck* prints error messages. Generally, if a severe problem exists with the root directory it is impossible to salvage the filesystem. *fsck* allows attempts to continue under some circumstances.

Phase 2 Error Messages

Possible error messages caused by problems with the root directory inode are shown below. The possible responses are discussed in the next section, "Phase 2 Responses."

`ROOT INODE UNALLOCATED. TERMINATING`

The root inode points to incorrect information. There is no way to fix this problem, so the command stops.

If this problem occurs on the Root filesystem, you must reinstall IRIX. If it occurs on another filesystem, you must recreate the filesystem using *mkfs* and recover files and data from backups.

`ROOT INODE NOT A DIRECTORY. FIX?`

The root directory inode does not seem to describe a directory. This error is usually fatal. The typical answer is yes.

`DUPS/BAD IN ROOT INODE. CONTINUE?`
: Something is wrong with the block addressing information of the root directory. The typical answer is yes.

Other Phase 2 messages have a REMOVE? prompt. These messages are:

`I OUT OF RANGE I=I NAME=F (REMOVE?)`
: A directory entry *F* has an inode number *I* that is greater than the end of the inode list. The typical answer is yes.

`UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F(REMOVE?)`
: A directory entry *F* has an inode *I* that is not marked as allocated. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. If the filesystem is not mounted and the **-n** option is not specified, and if the inode that the entry points to is size 0, the entry is removed automatically.

`DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE?)`
: Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory entry *F*, directory inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed. Typically, you should answer no the first time this error appears and yes the second time if you know the files claimed by the other inode.

`DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T FILE=F (REMOVE?)`
: Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. Typically, you should answer no the first time this error appears and yes the second time if you know the files claimed by the other inode.

Phase 2 Responses

Table A-2 describes the significance of responses to Phase 2 prompts:

Table A-2 Meaning of Phase 2 *fsck* Responses

Prompt	Response	Meaning
FIX?	n	*fsck* terminates.
FIX?	y	*fsck* treats the contents of the inode as a directory, even though the inode mode indicates otherwise. If the directory is actually intact, and only the inode mode is incorrectly set, this may recover the directory.
CONTINUE?	n	*fsck* terminates.
CONTINUE?	y	*fsck* attempts to continue with the check. If some of the root directory is still readable, pieces of the files system may be salvaged.
REMOVE?	n	Ignore the error condition. A “no” response is appropriate only if the user intends to take other action to fix the problem.
REMOVE?	y	Remove a bad directory entry.

Phase 3 Check Connectivity

Phase 3 of *fsck* locates any unreferenced directories detected in Phase 2 and attempts to reconnect them. It reports error conditions resulting from:

- unreferenced directories
- missing or full *lost+found* directories

Phase 3 Error Messages

Phase 3 has two types of error messages: information messages and messages with a RECONNECT? prompt. The possible responses are discussed in the next section, "Phase 3 Responses."

`UNREF DIR I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `(RECONNECT?)`

The directory inode *I* was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. The *fsck* command forces the reconnection of a nonempty directory. The typical answer is yes.

`SORRY. NO lost+found DIRECTORY`

No *lost+found* directory is in the root directory of the filesystem; *fsck* ignores the request to link a directory in *lost+found*. The unreferenced file is removed.

Use *fsck* **-l** to recover and remake the *lost+found* directory as soon as possible.

`SORRY. NO SPACE IN lost+found DIRECTORY`

There is no space to add another entry to the *lost+found* directory in the root directory of the filesystem; *fsck* ignores the request to link a directory in *lost+found*. The unreferenced file is removed.

Use *fsck* **-l** to recover and clean out the *lost+found* directory as soon as possible.

`DIR I=`*I1* `CONNECTED. PARENT WAS I=`*I2*

This is an advisory message indicating that a directory inode *I1* was successfully connected to the *lost+found* directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the *lost+found* directory.

Phase 3 Responses

Table A-3 explains the significance of responses to Phase 3 prompts:

Table A-3 Meaning of *fsck* Phase 3 Responses

Prompt	Response	Meaning
RECONNECT?	n	Ignore the error condition. This invokes the UNREF error condition in Phase 4. A "no" response is appropriate only if the user intends to take other action to fix the problem.
RECONNECT?	y	Reconnect directory inode *I* to the filesystem in the directory for lost files (*lost+found*). This may invoke a lost+found error condition if there are problems connecting directory inode *I* to *lost+found*. If the link was successful, this invokes a CONNECTED information message.

Phase 4 Check Reference Counts

This phase checks the link count information seen in Phases 2 and 3 and locates any unreferenced regular files. It reports error conditions resulting from:

- unreferenced files
- a missing or full *lost+found* directory
- incorrect link counts for files, directories, or special files
- unreferenced files and directories
- bad and duplicate blocks in files and directories
- incorrect counts of total free inodes

Phase 4 Error Messages

Phase 4 has five types of error messages:

- information messages
- messages with a RECONNECT? prompt
- messages with a CLEAR? prompt
- messages with an ADJUST? prompt
- messages with a FIX? prompt

The possible responses are discussed in the next section, "Phase 4 Responses." The typical answer is yes, except as noted.

`UNREF FILE I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `(RECONNECT?)`
: Inode *I* was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the **-n** option is omitted and the filesystem is not mounted, empty files are cleared automatically. Nonempty files are not cleared.

`SORRY. NO lost+found DIRECTORY`
: There is no *lost+found* directory in the root directory of the filesystem; *fsck* ignores the request to link a file in *lost+found*.

  Use *fsck* **-l** to recover and create the *lost+found* directory as soon as possible.

`SORRY. NO SPACE IN lost+found DIRECTORY`
: There is no space to add another entry to the *lost+found* directory in the root directory of the filesystem; *fsck* ignores the request to link a file in *lost+found*.

  Use *fsck* **-l** to recover and clean out the *lost+found* directory as soon as possible.

`(CLEAR)`
: The inode mentioned in the immediately previous UNREF error condition cannot be reconnected, so it is cleared.

`LINK COUNT FILE I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `COUNT=`*X* `SHOULD BE` *Y* `(ADJUST?)`
: The link count for inode *I*, which is a file, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed.

`LINK COUNT DIR I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `COUNT=`*X* `SHOULD BE` *Y* `(ADJUST?)`
The link count for inode *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed.

`LINK COUNT` *F* `I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `COUNT=`*X* `SHOULD BE` *Y* `(ADJUST?)`
The link count for *F* inode *I* is *X* but should be *Y*. The filename *F*, owner *O*, mode *M*, size *S*, and modify time *T* are printed.

`UNREF FILE I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `(CLEAR?)`
Inode *I*, which is a file, was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the **-n** option is omitted and the filesystem is not mounted, empty files are cleared automatically. Nonempty directories are not cleared. Typically, you should answer no the first time this error appears and yes the second time if you know the files claimed by the other inode.

`UNREF DIR I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `(CLEAR?)`
Inode *I*, which is a directory, was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the **-n** option is omitted and the filesystem is not mounted, empty directories are cleared automatically. Nonempty directories are not cleared. Typically, you should answer no the first time this error appears and yes the second time if you know the files claimed by the other inode.

`BAD/DUP FILE I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `(CLEAR?)`
Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. Typically, you should answer no the first time this error appears and yes the second time if you know the files claimed by the other inode.

`BAD/DUP DIR I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T* `(CLEAR?)`

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. Typically, you should answer no the first time this error appears and yes the second time if you know the files claimed by the other inode.

`FREE INODE COUNT WRONG IN SUPERBLK (FIX?)`

The actual count of the free inodes does not match the count in the superblock of the filesystem.

Phase 4 Responses

Table A-4 describes the significance of responses to Phase 4 prompts:

Table A-4 Meaning of *fsck* Phase 4 Responses

Prompt	Response	Meaning
RECONNECT?	n	Ignore this error condition. This invokes a CLEAR error condition later in Phase 4.
RECONNECT?	y	Reconnect inode *I* to filesystem in the directory for lost files (*lost+found*). This can cause a lost+found error condition in this phase if there are problems connecting inode *I* to *lost+found*.
CLEAR?	n	Ignore the error condition. A “no” response is appropriate only if the user intends to take other action to fix the problem.
CLEAR?	y	Deallocate the inode by zeroing its contents.
ADJUST?	n	Ignore the error condition. A”no” response is appropriate only if the user intends to take other action to fix the problem.
ADJUST?	y	Replace link count of file inode *I* with the link counted computed in Phase 2.
FIX?	n	Ignore the error condition. A “no” response is appropriate only if the user intends to take other action to fix the problem.
FIX?	y	Fix the problem.

Phase 5 Check Free List

Phase 5 checks the free-block list. It reports error conditions resulting from:

- bad blocks in the free-block list
- bad free-block count
- duplicate blocks in the free-block list
- unused blocks from the filesystem not in the free-block list
- total free-block count incorrect

Phase 5 Error Messages

Phase 5 has four types of error messages:

- information messages
- messages that have a CONTINUE? prompt
- messages that have a FIX? prompt
- messages that have a SALVAGE? prompt

The possible responses are discussed in the next section, “Phase 5 Responses.” The typical answer is yes.

`FREE BLK COUNT WRONG IN SUPERBLOCK (FIX?)`
: The actual count of free blocks does not match the count in the superblock of the filesystem.

`BAD FREE LIST (SALVAGE?)`
: This message is always preceded by one or more of the Phase 5 information messages.

Phase 5 Responses

Table A-5 describes the significance of responses to Phase 5 prompts:

Table A-5 Meanings of Phase 5 *fsck* Responses

Prompt	Response	Meaning
CONTINUE?	n	Terminate the command.
CONTINUE?	y	Ignore the rest of the free-block list and continue execution of *fsck*. This error condition always invokes a BAD BLKS IN FREE LIST error condition later in Phase 5.
FIX?	n	Ignore the error condition. A “no” response is appropriate only if the user intends to take other action to fix the problem.
FIX?	y	Replace count in superblock by actual count.
SALVAGE?	n	Ignore the error condition. A “no” response is appropriate only if the user intends to take other action to fix the problem.
SALVAGE?	y	Replace actual free-block bitmap with a new free-block bitmap.

Phase 6 Salvage Free List

This phase reconstructs the free-block bitmap. There are no error messages that can be generated in this phase and no responses are required.

Cleanup Phase

Once a filesystem has been checked, a few cleanup functions are performed. The cleanup phase displays advisory messages about the filesystem and status of the filesystem.

Cleanup Phase Messages

X `files` *Y* `blocks` *Z* `free`
: This is an advisory message indicating that the filesystem checked contained *X* files using *Y* blocks leaving *Z* blocks free in the filesystem.

`SUPERBLOCK MARKED DIRTY`
: A field in the superblock is queried by system commands to decide if *fsck* must be run before mounting a filesystem. If this field is not "clean," *fsck* reports and asks if it should be cleaned.

`PRIMARY SUPERBLOCK WAS INVALID`
: If the primary superblock is too corrupt to use, and *fsck* can locate a secondary superblock, it asks to replace the primary superblock with the backup.

`SECONDARY SUPERBLOCK MISSING`
: If there is no secondary superblock, and *fsck* finds space for one (after the last cylinder group), it asks to create a secondary superblock.

`CHECKSUM WRONG IN SUPERBLOCK`
: An incorrect checksum makes a filesystem unmountable.

`***** FILE SYSTEM WAS MODIFIED *****`
: This is an advisory message indicating that the current filesystem was modified by *fsck*.

`***** REMOUNTING ROOT... *****`
: This is an advisory message indicating that *fsck* made changes to a mounted Root filesystem. The automatic remount ensures that in-core data structures and the filesystem are consistent.