NAME

lfcheck − logical file system (LFS) consistency check and repair

SYNOPSIS

lfcheck [−synH] [lfs_name **]**

DESCRIPTION

Lfcheck audits UNIX logical file systems for consistency and corrects any discrepancies. Since these corrections will, in general, result in a loss of data, the program will request user concurrence for each such action. All questions should be answered by typing **yes** or **no** followed by a <CR>. Entering **yes** will cause the action specified by the question to take place, while a **no** response prevents the action from being taken. Entering **−y** or **−n** instead of yes or no has the same effect as setting the **−y** or **−n** option on the command line. Entering **?** when input is required prints a description of acceptable responses. Entering **!** allows the user to execute shell commands, and entering **quit** allows the user to exit the program (except during critical operations).

The options are entered as a dash (−) followed by the legal option characters in any order. If mutually exclusive options are present, the last specified (rightmost appearance in the character string) is the option taken.

Valid option characters are:

- **s** perform a salvage (repair) of the LFS
- **y** automatically answer all questions **yes**
- **n** automatically answer all questions **no**
- **H** print LFS header (configuration) information.

Note: **y** and **n** are mutually exclusive.

The program consists of six separate phases, some of which are skipped if they are not needed.

Phase one: *lfcheck* prompts for options and the LFS name if not specified on the command line, opens the filesystem as a UNIX file, reads and verifies the LFS header block, and initializes its tables.

Phase two: *lfcheck* examines the range of blocks belonging to each (allocated) logical file, checking for blocks which are outside the logical file system (BAD), belong to more than one logical file (DUP), belong to a file and overlap the overhead area of the logical file system (ALLOC & OVHD), or belong to a file but are not marked as allocated in the free-space bitmap on disk (ALLOC & FREE).

Phase three: is executed only if DUP blocks were found in phase two. This phase locates the first owner (logical file) of each DUP and adds the file to the faulty file list. The identity of the first file to own a block is not saved in phase two and must be determined by an additional pass over the file definition entries. A list of all duplicated blocks and all known owners is then printed.

Phase four: is executed only if the salvage option, **−s**, has *not* been specified (check-only) and compares the disk versions of the freelist and free-space bitmap to locate blocks included in the freelist but not marked free in the bitmap and vice-versa. A summary of the discrepancies is printed along with the number of entries that are used in the freelist.

Phase five: is executed only if the salvage option, **−s** , has been specified and loops through the faulty file list, printing the errors associated with each faulty file and prompting for corrective action. Correction usually consists of

deleting the logical file and freeing its blocks, though certain trivial errors can be corrected by *lfcheck* and the user is so informed and asked whether to *keep* the file instead of whether to delete it.

Phase six: is executed only when a salvage has been specified and builds a new freelist and bitmap based on the internal model of the LFS after corrections. During this phase, blocks that do not belong to any logical file and were not included in the freelist or free-space bitmap (MISSING) are automatically returned to the freelist. The new freelist and bitmap are written out to disk, and a summary of the error, allocated file, and free LF block counts is printed.

NOTE: All information on files and blocks is gathered before any alteration of the LFS is made.

EXAMPLES

In the following sample salvage run, the user's input is in bold print.

lfcheck

```
Options: ?
Valid options are any combination of:
        y − yes to all questions
        n − no to all questions
        s − perform LFS salvage
        c − perform LFS check (default)
        H − print LFS header
or <CR> − same as '−c'
Options: s
LFS Name: ?
A valid LFS name is xxx...xx
where /dev/xxx...xx is the name of the block device containing the LFS
LFS Name: lfs2
          /dev/lfs2
Phase 1: Verify Header and Initialize
Phase 2: Find Structural Errors
Phase 3: Locate First Owners
block =   130 owning lfn's =   2       1
block =   131 owning lfn's =   2       1
Phase 5: Delete Faulty Files
  >>>  Status: A=ALLOC, B=BAD, D=DUP, F=FREE, O=OVHD
lfn =     1     start =   128  size =     4    status = A.DF.
Delete file? ?
yes or no? yes
File deleted and blocks returned.
lfn =     2     start =   130  size =     3    status = ..D..
Delete file? quit
  >>>  Critical point in program -- can't quit now
Delete file? no
File kept.
lfn =     3     start =   400  size =    22    status = A..F.
TRIVIAL ERROR: Keep file? yes
File kept.
lfn =     5     start =  2290  size =    10    status = AB.F.
```

```
Delete file? yes
File deleted and blocks returned.
lfn =     6     start =    100  size =     2     status = A..FO
Delete file? yes
File deleted and blocks returned.
Phase 6: Build New Freelist
  >>>  Freelist contains 3 entries (2.36% full)

#alloc files= 5    (0.04%)      #faulty files= 5    (100.00%)
#dup LF blocks= 2    (0.09%) #dup LF block occurrences= 4
#free LF blocks= 2167 (94.26%)  #files retained= 2    (0.01%)
```

FILES

/dev/*lfs_name*	logical filesystem to be checked
/etc/lmtab	logical file mount table
stdout	DUP report; all reports, errors if redirected
stderr	user prompts and error messages

SEE ALSO

lfmount(1), lfumount(1), mklfs(1)

DIAGNOSTICS

Most error messages are self explanatory. Some errors, such as trying to check a mounted LFS or internal table overflow, require the user to decide whether to continue. In such cases, the decision to continue **may** result in incomplete or inaccurate results.

Lfcheck returns the following exit code values:

0 = normal termination (no LFS errors)
1 = LFS errors
2 = *lfcheck* internal error.

WARNINGS

Mounted logical file systems *cannot* be salvaged.

Checking a mounted logical file system may produce inaccurate reports.

Choosing to continue when *lfcheck* tables overflow during a salvage run prevents the addition of new entries to the overflowed table but does not cause the loss of any current entries. This may leave files on disk that have errors which cannot be detected in subsequent *lfcheck* runs once corrections have been made by the current run. If *lfcheck* were to simply exit on table overflow, however, then a partial repair could not be made and the LFS would have to be reloaded from a backup or rebuilt from scratch. A warning message regarding unrecorded errors is printed on program completion after overflow has occurred.

Once the user has begun to delete files, the option to enter **quit** in reply to a correction prompt is no longer allowed, and the **<rubout>** (interrupt) signal is disabled.

Should the user wish to run a check as a non-interactive process then a blanket reply (**−y** or **−n**) must be specified on the command line along with the LFS name, and output should be redirected.

BUGS

Currently, *lfcheck* does not recover from I/O errors and terminates abnormally when they are encountered.