LDOS (tm)LSI
UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #3

(c)1982

$29.95

Disk #3 Includes:

PCHECK/CMD - LDOS Directory Check Utility. Checks a drive or a file, (GAT, HIT, BOOT, etc.) for integrity.

PFIX/CMD - FIXES problem discovered by PCHECK/CMD. You may repair GAT or HIT table, repair BOOT sector, or repair FILE records. This program might fix that crashed disk that has you going nuts!

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but make sure it is SINGLE DENSITY!
2) BOOT the MASTER disk you received in this package in Drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density DATA diskette.

PFIX

The PFIX Utility is a comprehensive Directory Repair Utility. It allows the user to completely repair most directory problems, and also has the ability to transfer a faulty boot sector from a good disk. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable rigid disk systems. This program will locate and repair most inconsistencies in the directory information of a disk. In conjunction with PCHECK, most directory problems can be easily located and corrected without extensive knowledge of how directories are formatted. The syntax of the PFIX command is:

```
=======================================================
!  PFIX,:a,G,H,F,B=:d,L,P                               !
!                                                       !
!       :a = drive number to fix (colon optional)       !
!       G  = repair GAT table                           !
!       H  = repair HIT table                           !
!       F  = repair missing/invalid File data           !
!       B  = repair BOOT sector                         !
!       :d = take boot sector from this drive           !
!            (mandatory if B is specified)              !
!       L  = lockout cylinder above directory           !
!            (rigid systems only)                       !
!       P  = prompt for cyls, density, sides            !
!            (should be last parameter specified        !
!                                                       !
!       At least ONE parameter MUST be specified        !
!                                                       !
!  abbr: NONE                                           !
=======================================================
```

After selecting the desired parameters, the entire directory of the indicated disk is read into memory. If the P option is not specified, the program will use the information supplied by LDOS defining the diskette cylinder count, number of sides, and density. This information is located in the GAT sector of the diskette at format time. If it is possible that this particular information is invalid, use the P option, and you will be prompted to enter these parameters. If, for example, the entire GAT sector is zeroed out, LDOS will report this to be a xxxxxx density, single sided, 35 track disk. If the GAT is fixed using these parameters, and the disk is REALLY a 40 track disk, the corrections will be inaccurate.

If not enough memory is available, it will be reported, and the program will abort. If any errors occuring during the directory read, they will be reported, and the program will pause and allow you to select a R>etry, S>kip, or A>bort. Although it is possible to repair even an unreadable directory, unpredictable errors may be generated. For the normal operation of this program, it is assumed that all sectors of the directory are READABLE, even though they may be INCORRECT.

Each file of the directory is passed through a three stage process. In the first stage, the integrity of the files directory entry is fixed. In the second stage, the associated HIT table bytes are fixed. In the third stage, the associated GAT table bytes are fixed. When all specified files have been checked, and all errors have been corrected, the corrected data will be written back to disk per the command parameters. Either the GAT, HIT, file records, and/or BOOT sector will be updated to the disk. Although the entire directory is repaired in memory, only the requested data to be fixed is written back out to the disk. The file PFIX/CMD is capable of fixing most errors that may be reported by PCHECK.

If only the GAT and/or HIT are specified, and an error is found in any of the files directory records, the fix will abort and a message will be reported to use the "F" option. This is because it would be unreliable to repair a GAT or HIT when the root information is invalid. This is to prevent a user from inadvertently destroying any relevant data.

The following is a list of the errors that PCHECK can fix. A technical section will follow that will describe the logic used by PFIX, and probable causes for the error conditions.

1. Cylinder xxx has an invalid GAT table byte
2. HIT byte at xxH invalid or extraneous
3. Filename contains non-ascii characters
4. End of File Sector beyond allocated sectors
5. No terminator for extent field
6. Directory links to record not linking back to it
7. Track assigned that is beyond diskette boundary
8. Extension assigned before end of extents
9. Forward link to inactive entry
10. Forward link to non-extension entry
11. Extension record not assigned to any files
12. Multiple files assigned to single granule
13. Directory record has invalid HIT byte
14. Directory record has a zero HIT byte
15. Extended directory record has invalid HIT byte
16. Extended directory record has a zero HIT byte

This section contains the logic that PFIX uses to repair the directory. The novice user need not be concerned with this information as it is offered only as technical reference for the advanced user.

Pass #1

Check to be sure that all 11 bytes of the filename contain characters within the range of 20H to BFH (printable ascii characters). If any are found to be outside this range, replace with an "X". (If this occurs, a HIT repair will be required to access this file from DOS)

The 5 extents of the associated file record are each checked on a sector by sector basis.

If any extent points to a track that is beyond the diskette boundary, enter a terminator (FFH) at the current location.

If any granules are indicated in a file's extents, and the corresponding GAT bit is set (assigned to a previous file), the record will be terminated at the current location.

If all 5 extents have been parsed, and no terminator found (FEH or FFH) then enter a terminator at the last position.

If an extended record is found, and it is not at the 5th position in the extent field, ignore and continue.

If the extended record is found, locate its position in the directory. If bit 4 of the first byte in the record is not set (inactive file), delink the extension. If bit 7 is not set (not an extension), delink the extension. The second byte in the entended record is a link back to the previous record. If this byte is incorrect, delink the extension.

The de-linking of the files is to protect against possible overlaying another file that does not in fact link back to the primary. By de-linking a file, data past the de-link will be lost, but the data will not accidentally be force-linked to an incorrect entry.

This process continues until an FFH terminator is reached, or a terminal error is found and a terminator is forced.

When processing is completed, the number of sectors assigned to the file is calculated and compared to the end of file sector in the directory record. If the end of file is larger than the allocated records, then force the number of allocated records into the end of file sector.

Pass #2

This pass constructs the HIT table. Each file is traced from its primary directory entry through all extents until an FFH terminator is reached. The entire HIT table will be re-constructed starting with a page (256 bytes) of zeroes. All active primary and extended directory records are logged correctly into this table.

Pass #3

This pass constructs the GAT table. As with pass 2, each file is traced through all of its extents. The entire GAT table will be re-constructed starting with all null characters. All active primary and extended directory records are logged correctly into this table.

Sometimes, one type of an error will inherently cause another type of error. In these cases, fixing even one error may correct several others. As an example, if there is no terminator for a record, repairing this will sometimes correct a multiple files assigned to a single granule error.

When using this utility on Rigid drives, there is one error that may be repaired that the user should be made aware of. LDOS sets aside an extra track right next to the directory, and shows this track as allocated in the GAT table. This track is used by the Laredo system as overhead, and should not be used. The PFIX utility will de-allocate this granule, as it is not assigned to any files. This is only on the Laredo system, and cannot be detected via the DCT parameters. Due to this situation, PFIX will allocate this track if the following conditions are met:

The drive being fixed must be a Rigid drive.

The GAT fix option must have been specified.

The GAT entry for the cylinder immediately past the directory must be a null byte (not allocated to any files) AFTER the GAT re-build.

The L option must have been specified.

If these conditions are met, then the extra track is automatically allocated. This could essentially lock-out an available track on a non-Laredo system, but will cause no other problems. If you are not sure if your rigid system uses this cylinder or not, then use the L option. It is better to lock out a cylinder than crash the system.

PCHECK

The PCHECK Utility is a comprehensive Directory Check Utility. It allows the user to completely check the integrity of disk storage systems. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable rigid disk systems. This program will locate and report any inconsistencies in the directory information of a disk. In conjunction with PFIX, most directory problems can be easily corrected without extensive knowledge of how directories are formatted. The syntax of the PCHECK command is:

```
=====================================================
!  PCHECK,*,:a                                        !
!  PCHECK,*,filespec                                  !
!                                                     !
!       :a = drive number to check (colon optional)   !
!       filespec = any valid LDOS filespec            !
!                                                     !
!       an asterisk (*) preceeding the command will   !
!       send the output to the video AND printer      !
!                                                     !
!  abbr: NONE                                         !
=====================================================
```

After selecting the mode of operation (full disk or file), the entire directory of the indicated disk is read into memory. If not enough memory is available, it will be reported, and the program will abort. If any errors occuring during the directory read, they will be reported, and the program will pause and allow you to select a R>etry, S>kip, or A>bort. Although it is possible to check even an unreadable directory, unpredictable errors may be reported. For the normal operation of this program, it is assumed that all sectors of the directory are READABLE, even though they may be INCORRECT.

Each file specified is passed through a three stage check. This will either be the single file specified, or all files if an entire disk is specified. In the first stage, the integrity of the files directory entry is checked. In the second stage, the associated HIT table bytes are checked. In the third stage, the associated GAT table bytes are checked. When all specified files have been checked, and all errors have been reported, a total error count will be issued and the program will exit back to DOS. The file PFIX/CMD is capable of fixing most errors that may be reported by PCHECK.

The following is a list of the possible error messages that PCHECK may report. A technical section will follow that will describe the logic used by PCHECK, and probable causes for the error conditions.

1. Cylinder xxx has an invalid GAT table byte
2. HIT byte at xxH invalid or extraneous
3. Filename contains non-ascii characters
4. End of File Sector beyond allocated sectors
5. No terminator for extent field
6. Directory links to record not linking back to it
7. Track assigned that is beyond diskette boundary
8. Extension assigned before end of extents
9. Forward link to inactive entry
10. Forward link to non-extension entry
11. Extension record not assigned to any files
12. Multiple files assigned to single granule
13. Directory record has invalid HIT byte
14. Directory record has a zero HIT byte
15. Extended directory record has invalid HIT byte
16. Extended directory record has a zero HIT byte

This section contains the logic that PCHECK uses to determine the correctness of the directory. The novice user need not be concerned with this information as it is offered only as technical reference for the advanced user.

During passes 1-3, if an error is found, it is reported in the following format:

```
filename/ext        @ Directory Sector xx, Relative Byte xxH.
"error message string"
```

If a non ascii character is found in the filename, its position will be highlighted with a graphic block. In all errors reported in the above format, the directory location will refer to the PRIMARY directory record, regardless of where the error occured along the file.

Pass #1

Check to be sure that all 11 bytes of the filename contain characters within the range of 20H to BFH (printable ascii characters). If any are found to be outside this range, report error 3. The offending characters are displayed as a graphic block to aid in identifying their positions.

The 5 extents of the associated record are each checked on a sector by sector basis.

If any extent points to a track that is beyond the diskette boundary, report error 7.

If all 5 extents have been parsed, and no terminator is found (FEH or FFH) then report error 5 and terminate.

If an extended record is found, and it is not at the 5th position in the extent field, report error 8 and continue.

If the extended record is found, locate its position in the directory. If bit 4 of the first byte in the record is not set (inactive file), report error 9 and terminate. If bit 7 is not set (not an extension), report error 10 and terminate. The second byte in the entended record is a link back to the previous record. If this byte is incorrect, report error 6 and terminate.

This process continues until an FFH terminator is reached, or a terminal error is found.

When processing is completed, the number of sectors assigned to the file is calculated and compared to the end of file sector in the directory record. If the end of file is larger than the allocated records, report error 4.

Pass #2

This pass checks the integrity of the associated HIT byte for the current file. Each file is traced from its primary directory entry through all extents until an FFH terminator or a terminal error is reached.

If the primary directory entry contains an invalid HIT byte, report error 13 and continue. If the primary directory entry contains a zero HIT byte, report error 14 and continue.

If an extended directory record contains an invalid HIT byte, report error 15 and continue. If an extended record contains a zero HIT byte, report error 16 and continue.

Pass #3

This pass checks the integrity of the associated GAT bytes for the current file. As with pass 2, each file is traced through all of its extents.

If the current granule has already been allocated (either by itself or another file), report error 12 and continue.

NOTE: Only a single error 12 will be reported for a single file even though there may be several occurances of it.

When these steps have all been completed, and a single file has been specified, the number of errors will be reported and an exit will be made back to DOS. If an entire disk is specified, 3 additional checks are made on the directory.

Check #1

When Pass #1 is made in the above process, it checks every directory record that has bit 4 set and bit 7 NOT set (active primary directory record). As the current file is being checked, bit 5 of all the associated directory primary and extended records is set to indicate a completed file. During the Check #1 phase, the entire directory is re-scanned. If any files have bit 4 set, and not bit 5, then bit 7 MUST be set and error 11 is reported. This condition indicates an extended directory record that has not be linked to any primary record (an extraneous directory record). This condition could result if one of the above passes terminates on a fatal error and an extension is not checked. If no terminal type errors are reported, but error 11 is issued, then there is an extra record in the directory that is not associated with any files.

Check #2

As the program processes pass #2, it completely builds a second HIT table in another buffer. During the check #2 phase, the computed table is compared byte for byte with the HIT table as read from the diskette. If any of the bytes do not match, report error 2 and the relative byte of the mismatch. If none of the pass #2 errors are reported, but error 2 is issued, then there is an extra byte in the HIT table at the corresponding location.

Check #3

As the program processes pass #3, it completely builds a second GAT table in another buffer. Before check 3 is made, the disk lockout table is overlaid onto the built GAT table so that locked out tracks will not be counted. During the check #3 phase, the computed table is compared byte for byte with the GAT table as read from the diskette for as many tracks as there are on the disk. If any of the bytes do not match, report error 1 and the relative cylinder of the mismatch. If none of the pass #3 errors are reported, but error 1 is issued, then there are extra grans allocated on the disk that are not assigned to any files.

Sometimes, one type of an error will inherently cause another type of error. In these cases, fixing even one error may correct several others. As an example, if there is no terminator for a record, chances are that error 12 will be reported also. If a terminal error is found in a directory link to an extension, error 2 may also be reported.

When using this utility on Rigid drives, there is one error that may be reported that should be considered normal. LDOS sets aside an extra track right next to the directory, and shows this track as allocated in the GAT table. This track is used by the Laredo system as overhead, and should not be used. PCHECK may report an invalid GAT Table Byte at this track location, and it should be considered normal. The Rigid drive user should be aware of this problem, especially as it relates to the PFIX utility, which may de-allocate this track, or allocate it to an unassigned file.

LDOS (ta)LSI UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #2 (c)1982

$29.95

Disk #2 Includes:

PMOD/CMD - A Disk-File-Memory Modification Utility.

PMOD,aaaa (display memory address)
PMOD,filespec,aaaa (file to display,starting rel. sector)
PMOD :b,c,d (drive,cylinder,sector to display)

UTILITY DISK #3 (c)1982

$29.95

Disk #3 Includes:

PCHECK/CMD - LDOS Directory Check Utility. Checks a drive or a file, (GAT, HIT, BOOT, etc.) for integrity.
PFIX/CMD - FIXES problem discovered by PCHECK/CMD. You may repair GAT or HIT table, repair BOOT sector, or repair FILE records. This program might fix that crashed disk that has you going nuts!

UTILITY DISK #4 (c)1982

$29.95

Disk #4 Includes:

PFIND/CMD - FIND a string, replace a string, find a byte, find a WORD, replace a byte or word! FIND ANYTHING on the disk FAST! Replace it FAST!
PCOMPARE/CMD - Compare a FILE or a sector to any other file or sector. Detects the minutest differences! Send the outputs from above to video or printer.

UTILITY DISK #5 (c)1982

$29.95

Disk #5 Includes:

PREFORM/CMD - Disk Re-Formatter Utility. Fixes CRC errors and "stengthens" the format on older disks. A FORMAT without ERASE utility(5" floppy only)
PVU/CMD - a Disk Verification Utility. Detects bad sectors. Fix them with PREFORM/CMD.
PERASE/CMD - Disk Bulk Eraser (5" floppy only)

UTILITY DISK #6 (c)1982

$29.95

Disk #6 Includes:

PCLEAR/CMD- Erases unassigned sectors and Directory slots. Removes ALL traces of "killed" files!
PSS/CMD - Sector Status Utility. Finds WHAT file is assigned to WHAT sector.
PMAP/CMD - Disk-File Mapping Utility. Map a FILE or a disk! Shows exactly WHERE all files are with extents.

UTILITY DISK #7 (c)1982

$29.95

Disk #7 Includes:

PMOVE/CMD - A SUPER FAST Multiple Copy Utility!
PDIRT/CMD - Read a Mod III TRSDOS directory from LDOS!
PASSGO/CMD - Removes passwords from A file or a WHOLE disk!
PUN/CMD - UN-REPAIR a disk. (Mod I only)
PEX/CMD - A disk exerciser for head cleaning kits.

UTILITY DISK #8 (c)1982

$29.95

Disk #8 Includes:
PMX/FLT-MX80 graphics
PHELP/CMD -Instant HELP! Very complete. Saves time!
PBOOT/FIX - Customize YOUR LDOS Boot graphics and message!
PFILT/FLT - A User Definable Filter. Very flexible!
DVORAK/FLT - DVORAK/QWERTY Keyboard Filter.
DVORAK/JCL - DVORAK Keyboard Table (used with PFILT/FLT)
CODE/JCL - Keyboard EN-Coding TABLE (used with PFILT/FLT)
DECODE/JCL - Keyboard DE-Coding Table (used with PFILT/FLT)