UTILITIES

BACKUP

The backup utility allows a directory and any sub-directories it might contain to be copied from one disc to another. This is particularly useful for backing up a disc - hence the name.

DESRIPTION

The syntax for the command is:

```
     +++BACKUP <source_drive> <destination_drive>
```

Before backup is invoked, however, it is first necessary to set both the source and destination drives into the correct directories using either CHGDIR or SETDIR.

Whenever backup encounters a sub-directory on the source drive, it creates an identical sub-directory on the destination drive and copies all files from the source sub-directory to the destination sub-directory.

CHGDIR

CHGDIR is used to move to a different directory. It is a shorthand version of SETDIR and can move in three directions - laterally, down to the next lowest level, or back up to a higher level. The syntax of CHGDIR makes moving to adjacent directories faster than SETDIR.

DESCRIPTION

The syntax is:

```
     +++CHGDIR <
     +++CHGDIR ^NEWDIRECTORY
     +++CHGDIR >NEWDIRECTORY
     +++CHGDIR >2.NEWDIRECTORY
     +++CHGDIR <0
```

The first example moves the directory on the current work drive to the next higher directory level.

The second example moves the directory on the current work drive to a new directory at te same level.

The third example moves the directory on the current work drive to the next lowest level.

The fourth example works the same as example three except that it works on drive 2.

The last example moves the current directory on drive zero up one level.

COPY

The COPY command is identical to the standard FLEX copy command except that files can be copied from other directories on the disc to the current directory. It is not possible to copy a file from the current directory to another directory - this can be done by setting oneself into the other directory and copying from the one you were originally in.

DESCRIPTION

The general syntax is identical to the FLEX copy command with the addition of a directory path in the source file.

```
     +++COPY <source file spec> <destination file spec>
     +++COPY <source file spec> <drive>
     +++COPY <source file spec> <drive> <<match list>>
```

Only the source file spec can accept a directory path.

The first example allows a single file to be copied from one directory to another. For example:

```
     +++COPY 1.[HOME.FRED.JIM]GAMES.TXT LIFE.TXT
```

copies the file GAMES.TXT in the directory JIM on drive 1 to a file called LIFE.TXT in the current directory on the work drive. Similarly

```
     +++COPY 1.[HOME.FRED.JIM]GAMES.TXT 0.LIFE.TXT
```

copies the file to the current directory on drive 0.

Directory paths need not be specified.

```
     +++COPY GAMES.TXT LIFE.TXT
```

would copy the files in the same directory on the work drive, and

```
     +++COPY GAMES.TXT 0.LIFE.TXT
```

would copy the file from the current directory on the work drive to the current directory on drive 0.

To specify a directory which is a sub-directory of the current one, the following syntax can be used:

```
     +++COPY [.JON]TIC-TAC.TXT TIC-TAC
```

Assuming the current directory is [HOME.JIM] then the file would be copied from [HOME.JIM.JON] on the work drive to the current directory on the work drive.

The second example of copy is used to copy a file from one directory to another while still retaining the original name:

```
     +++COPY 1.[HOME.FRED.JIM]LIFE.TXT 1
```

would copy LIFE.TXT from the JIM directory on drive 1 to the current directory on drive 1. NOTE that the destination will not default to the work drive and must be specified.

The third example of copy is used to copy from either the current directory, or a specified directory on the source drive to the destination drive. If a match list is also specified only matching files will be copied.

```
     +++COPY 1.[HOME.FRED.JIM] 0 .TXT
```

would copy all fiels with the extension .TXT in the directory JIM on drive 1 to the current directory on drive 0.

LIMITATIONS

Copy cannot be used to copy files with the extension .DIR because this does not preserve the backpointer contained in the directory. Any attempt to do this will be flagged as an error.

NOTE:

It is of course possible to rename a .DIR file to another extension, copy it and rename it back again.

This is not advised because the backpointer, which is used by several HIER utilities, will be corrupted. This can cause upredicatble results and even loss of files.

DELDIR

DELDIR is used to delete a directory from the current directory. All files in the specified directory, and any sub-directories which it might contain are deleted before the directory itself is finally deleted. DELDIR will prompt the user before deleting a file but will automatically delete an empty directory..

Files which have been delete or write protected using the FLEX PROT utility will not be deleted unless the -p option is used.

DESCRIPTION

The general syntac for the command is:

```
     +++DELDIR directory_name <-p>
```

Normally DELDIR will prompt the user for each file deletion, in a manner similar to the FLEX DELETE utility. This can, however, become rather tediioys if the directory being deleted holds a large number of files or sub-directories, and so the -p option allows all files to be deleted without prompting.

Because the -p option is rather powerful, DELDIR tries three times to discourage the user from making a mistake before proceeding to delete all the files.

Should the user decide to to continue with the -p option, DELDIR will revert to prompting for each deletion.

A directory will only be deleted if all the files it contains have been deleted, including any sub-directories and their files.

For example if DIR produced the following

```
Directory of drive 3

Disc: GAMES    #2                  Directory JIM

First free sector 0B 05            Last free sector 4C 0f

 Name    Ext   Prt   Start   End   Size    Type    Date

LIFE    .TXT         01 01  03 04    27      00    2-7-84
GAMES   .DIR     D   03 05  03 08     4      00    31-12-83

216 Free Sectors
```

then:

```
     +++DELDIR GAMES -p
```

would delete alll the files in the GAMES directory along with any files in sub-directories under GAMES, after prompting:

```
     Delete all files with no further prompting ?
     Even write protected files will be deleted - continue ?
     Are you sure ?
```

if the user decided against the -p option by answering any prompt with anything but Y, DELDIR prints the following message before continuing:

```
     You will be prompted for each deletion
```

DELDIR prompts for deletion in the following way

```
     Delete "3.[HOME.JIM.GAMES]TIC-TAC.TXT ?
     Are you sure ?
```

Answering Y to each prompt will delete the file.

DIR

DIR is an extension of the standard FLEX CATalog utility. The enhancements include the ability to specify a directory path, to optionally list deleted files and a more informative display.

DESCRIPTION

The general syntax for the command is

```
     +++DIR <<+D>> <<drive list>> <<mach list>>
```

The +D option causes DIR to include all deleted files. Drive list includes the optional directory path and the match list defines the types of files DIR is to list.

```
     +++DIR 1.[.FRED.JIM] .TXT
```

would list all the files in the sub-directory JIM below the current directory on drive 1 with the extension .TXT

```
     +++DIR +D 1..TXT
```

would list all files on drive 1 with the extension .TXT, including any deleted files whose directory entries had not been reused.

The format of the display on the terminal has been enhanced to include the directory name, the first and last free sectors on the disc, and more file related information.

For example

```
     +++DIR
```

would produce

```
Directory of drive 1

Disc: GAMES    #2                  Directory JIM

First free sector 0B 05            Last free sector 4C 0f

 Name    Ext   Prt   Start   End   Size    Type    Date

LIFE    .TXT         01 01  03 04    27      00    2-7-84
GAMES   .DIR     D   03 05  03 08     4      00    31-12-83

216 Free Sectors
```

where

Start is the starting track and sector of the file in hex.

End is the ending track and sector in hex.

Size is the decimal number of sectors the file occupies.

Type shows the file typw - usually 00 for an ordinary file and 02 for a random file.

Date is the file creation date.

DISCCHK

The disc check utility DISCCHK is used to check the entire structure of the specified disc. It does this in the folling manner:

1). Starting at the HOME directory and continuing through all the sub-directories, each file is opened and its sectors read to make sure they are not damaged. As each sector is read its sequence number is the file is checked; any out-of-sequence sectors are flagged to the user. Finally the last sector of the file is checked against its last sector entry in the directory.

2). As each sector of a file is read it is marked as used in a table maintained by DISCCHK. Should another file subsiquently claim to own the same sector, DISCCHK reports an error and marks the disc as UNSTABLE; meaning that the free chain cannot be re-linked until the clash has been rectified.

3). Once DISCCHK has built the table of used sectors it gives the user the option to re-link the free chain.

4). Processing the free chain is done in two parts. First DISCCHK assumes that all sectors not in its allocation table are either defective or they belong in the free chain. It therefore reads all the unallocated sectors in its table to see if they are defective. Any defective sectors are marked as used in the table. At the end of this phae DISCCHK reports the number of sectors it beleives should be in the free chain.

5). Finally DISCCHK asks the user if he / she wishes to re-link the chain. If the user replies Y and there have been no multiply allocated sectors, DISCCHK uses its table to relink the free chain into ascending order and updates the system information record.

DESCRIPTION

The syntax is for the command is:

```
+++DISCCHK <drive number>
```

MESSAGES

DISCCHK produces a number of error and informative messages as it processes the disc.

Informative messages:

```
Processing home directory
Processing files
```

These messages tell the user where in the sequence DISCCHK is.

Warning messages:

```
SIR read failure, code <n>
```

This indicates that DISCCHK was unable to read the system information record to obtain the necessary data to start processing the disc. The error number is the one returned by the FLEX FMS.

```
Sector read failure - error <n> , track <aa> sector <bb>
```

While processing the disc, DISCCHK was unable to read a sector from the disc. This error is not fatal if the sector is in the free chain.

```
Directory CLASH at track <aa> sector <bb>
```

A sector in the home directory has been allocated more than once. This error will not cause DISCCHK to terminate but may show up as other errors later on.

Insufficient memory to run DISCCHK

This message can means that either the system has too little memory to run or that the system information record on the disc is corrupted. DISCCHK uses the highest track and sector information from the system information record to allocate enough memory for the sector useage table. 16Kb of memory starting at 0000 is enough to check a 5 Mb drive.

Last sector incorrect in directory of file <filenmae>

The track and sector that DISCCHK determined were the last in the file do not agree with the directory entry.

Warning orphaned directory - <directory name>

This message means that the backpointer in the directory <directory name> does not point to a parent directory or the home directory. In order to ensure that no files are lost the directory concerned should be copied using BACKUP which will restore the backpointer.

<directory name><filename> SEQUENCE ERROR track <aa> sector<bb>

This message means that the sequence number contained in bytes 2 and 3 of the sector are not the next in the logical sequence. This error is normally an early indication that there will be a sector clash with another file. This error is not fatal but the file should be checked, if possible, to ensure that it contains the correct information.

<directory name><filename> - UNSTABLE

DISCCHK has decided that the file is unstable, often due to a sector read failure.

<directory name><filename> CLASH at track <aa> sector<bb>

This message indicates the the file named claims to own sectors which DISCCHK beleives belong to a previously processed file. This error will prevent the free chain from being relinked and the offending file or files must be deleted. Normally if the files are deleted and the chain is not re-linked the disc will remain in an unstable condition.

ERROR - setbit called with a sector larger than disc size.

DISCCHK has tried to mark as allocated a sector which is larger than the system informatio record claims the disc will hold.

sub-directory overflow

DISCCHK currently has enough room to process 300 sub-directories, and the current disc contains more than 300.

HOME

HOME is used to return the specified disc to the HOME directory
It is VERY IMPORTANT that ALL discs are HOMED before they are used

DESCRIPTION

The syntax is:

```
     +++HOME <drive>
```

HOME will report an error if it is unable to update the system information record, or the first directory sector, track 0 sector 5.

LIST

The LIST command is identical to the FLEX utility of the same name, except that a directory path can be specified before the filename.

DESCRIPTION

The general syntax is:

```
     +++LIST <file spec> <<line range>> <<+options>>
```

The file spec can include a directory path if required, otherwise it will default to the current directory.

```
     +++LIST [HOME.FRED.JIM]LIFE.TXT
```

would list the file LIFE.TXT in the directory JIM on the current work drive.

The line number range and options are the same as the standard FLEX LIST.

MAKEDIR

MAKEDIR is used to create a sub-directory in the current directory on the specified drive. The directory is delete protected to prevent the accidental erasure of valuable files. A directory path is not permitted with this command.

DESCRIPTION

The syntax for the command is:

```
     +++MAKEDIR JIM
```

This will make a new directory called JIM (a file called JIM.DIR) in the current directory on the work drive.

```
     +++MAKEDIR 2.FRED
```

This will make a neq directory called FRED (a file called FRED.DIR) in the current directory on drive 2.

If the directory already exists, MAKEDIR will report the error and exit.

MAKEDIR will initially allocate four sectors to te new directory, which is enough room for 40 files. FLEX will automatically allocate additional sectors to the directory if more than 40 files ae created.

MOVE

MOVE is used to move a file, or directory from one node in the HIER file structure to another.

DESCRIPTION

The syntax for MOVE is:

```
     +++MOVE <source> <destination>
```

for example:

```
     +++MOVE [HOME.GAMES]LIFE.TXT [HOME]
```

The first parameter defines the path to the file to be moved, and the second parameter defines the new path to the file.

A drive number can be specified as follows:

```
     +++MOVE 2.[HOME.GAMES]LIFE.TXT [.JIM]
```

in this case LIFE.TXT in the GAMES directory on drive 2 is moved to the JIM directory below the current directory on drive 2.

To move a file into the working directory only the drive number need be specified:

```
+++MOVE [HOME.GAMES]LIFE.TXT 1
```

This assumes that the current work drive is drive 1, and the working directory is not GAMES.

Finally MOVE can change the name of the file as it is moved:

```
     +++MOVE [HOME.GAMES]LIFE.TXT [.JIM]OLDLIFE.TXT
```

Note that all file names must include both the name and the extension, and files cannot be MOVEd from one disc to another.

RUN

RUN is used to activate a .CMD file in another directory on the disc. It comes in two forms RUN.CMD and RUN.LOW. The .LOW version is used to run files which reside in te FLEX utility command area.

DESCRIPTION

The syntax for RUN is:

```
     +++RUN [HOME.BACKUP]BACKUP
```

This will activate the file BACKUP.CMD in the directory BACKUP.

If the file runs in the FLEX utility area the syntax is:

```
     +++RUN.LOW [HOME.BACKUP]BACKUP
```

SETDIR

SETDIR is used to set the current working directory on a given disc.

DESCRIPTION

The syntax for SETDIR is:

```
     +++SETDIR [HOME.FRED.JIM]
     +++SETDIR 2.[HOME.FRED.JIM]
     +++SETDIR [.JIM]
```

The first example sets the current directory on the work drive to JIM along the path FRED from the HOME directory.

The second example does the same thing for drive 2.

The third example assumes that the current directory contains a sub-directory called JIM.

ERROR MESSAGES

SETDIR will give an error:

```
     Directory specification error
```

if the directory could not be found. In this case the current directory remains as it was before SETDIR was invoked.

TREE

The TREE utility prints a listing of all the directories and files on a disc. Each sub-directory level is indented two spaces from the left margin.

DESCRIPTION

The syntax is:

```
     +++TREE <drive>
```

Starting at the home directory and proceeding through all the directories, TREE will list all the files on the chosen disc.

UNDER

UNDER is used to determine the number of sectors, files and directories under a given node on the disc. This can be important when using BACKUP since it allows the user to determine if there is enough room on the destination disc.

DESCRIPTION

The syntax is:

```
     +++UNDER <drive>
```

and is issued when the user is at the desired directory node.

UNDER will then search all the files under the specified node and give the following response:

```
sectors: <nnn>, files: <nnn>, directories: <nnn>
```

each <nnn> is in decimal.

For example, if the user is in directory FRED, then UNDER would search FRED an JIM and print out

```
sectors: 125, files: 3, directories: 1
```

WHERE

WHERE prints the path to the current directory on the disc.

DESCRIPTION

The syntax is:

```
     +++WHERE <<drive>>
```

If a drive is not specified the current work drive is used.

A typical output from WHERE is

```
     [HOME.JIM.GAMES]
```

APPENDIX