The PC Emulator Survival Guide (Part 8)

by Gordon Gilmore

This month we take a look at DOS file security on the PC. The most obvious security concern at present - viruses - we're not going touch on at all. Instead we are going to look at commands within DOS to help us keep our data and files in good order.

VERIFY

If you click your menu button over a RISC OS disc drive icon you will see a menu item Verify. Selecting this item scans through every sector on the disc and reports whenever a bad sector is discovered. When you format a disc to an ADFS format, the disc is automatically verified and any bad sectors are mapped out

As with many things, DOS is not as smart as it might be. The DOS VERIFY command does not check discs for integrity. It merely checks that a file can be read back with no errors, and it does this when that file is being written to disc. It does not scan every sector on a disc like the ADFS equivalent. Typing VERIFY ON will switch the facility on, and obviously VERIFY OFF will turn it off again. If you are not sure whether it's on or off, typing VERIFY alone will tell you. The price, for the peace of mind of knowing that the data has been saved properly, is a longer save time. This might not be noticed for short files but might be a problem for longer ones. This peace of mind, however, is limited since VERIFY does not check that the data can be recovered from the sector it has just been saved to.

Unfortunately, although the ADFS Verify command will detect all the bad sectors on a DOS disc, it will not map them out for you. If you really do need a cast iron check on your discs you will have to buy one of the many utility programs which provide full verify and map out facilities (the one I use is an ancient version of PCTools).

Although using the VERIFY command will pick up some faulty saves, others may be missed, and in any case, floppy discs do from time to time sprout a bad sector or two spontaneously. So what can you do if you try to read a file and you get the message:

Sector not found
error reading drive A
Abort, Retry,
Ignore, Fail?

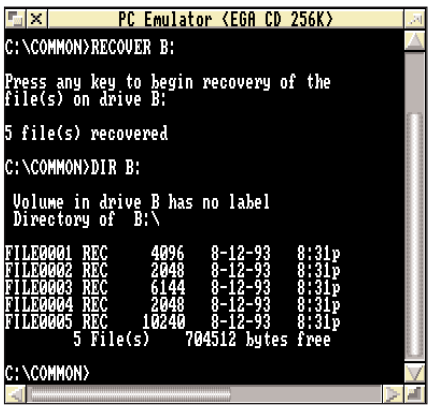

Figure 1. Use of RECOVER to rebuild a disc directory

First - don't panic! All may not be lost. There is always a slight chance that pressing R, to try again, will do the trick. If that doesn't work then you might press I. The bad block would be ignored, and hopefully the remainder of the file would be read. The result will be an incomplete file but if it contains only text, the recovered portion may be better than nothing. The missing portion could easily be typed in again. A missing block from a binary file, such as a program, is usually fatal and you will have to resort to your backup. (What no backup! - read on.)

The F, for Fail response to a bad read operation would allow a succession of commands to continue after ignoring the current command altogether. A useful tip in this context is that it may be better to abort, A, rather than fail a faulty read operation. Suppose you are updating a file FILE.EXT on your hard disc by copying from floppy:

```
COPY A:FILE.EXT
```

If the read fails and you abort the command you may find your original copy still intact. Failing, the command will delete the original hard disc file on the assumption that it holds invalid data.

RECOVER

Having found a bad sector on a disc it is as well to deal with it as soon as possible, and for this purpose we have the DOS command RECOVER. There are two ways in which you can use this command: on a file-by-file basis, or to recover complete directory data. However, before launching into that it would be a good idea to copy as many of the files from the corrupted disc to a new disc before starting the recovery process. For a floppy disc that would be easy; for a hard disc that really means a complete backup to floppies. Whatever else you do, it would be a good idea to make a copy of the disc directory either as hard copy or a copy on file (not on the dodgy disc!) for future reference.

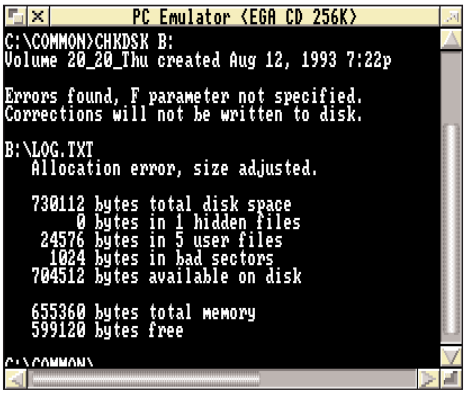

Figure 2. CHKDSK used after RECOVERy of a file

Now let s pretend that I have a file called LOG.TXT on a floppy disc in drive B which I know to have a bad block in it. I recover the file by specifying the filename in the command as in the following dialogue:

```
C:\COMMON>RECOVER B:LOG.TXT

Press any key to begin recovery of the
file(s) on drive B:


4096 of 5120 bytes recovered

C:\COMMON>DIR B:

 Volume in drive B is 20_20_Thu
 Directory of          B:\

LOG      TXT     5120   8-09-93   3:48p
PROG     DAT     1678   3-17-87  12:00p
PROG2    EXE     5825   3-17-87  12:00p
PROG     COM     1561   3-17-87  12:00p
PROG2    SRC     9529   3-17-87  12:00p
  5 File(s)     704512 bytes free

    C:\COMMON>
```

RECOVER has copied as much of LOG.TXT as possible to a new location on the disc and marked the bad sector for future reference. Now, because of the bad sector, there must be chunk of data missing. RECOVER told us that it had recovered 4096 of the 5120 bytes it was expecting but the directory entry still indicates the full 5120 bytes. This means that the directory information and the file allocation table (the FAT - which is used to keep track of which blocks on the disc belong to which file) no longer match.

We can put this right by using CHKDSK but before we do this we will look at the second use of RECOVER. This is a pretty disruptive operation and should not be undertaken lightly. Normally it would be much quicker and easier to restore the files from your backup copy. This form of the RECOVER command, specifying only the drive letter, is intended to rebuild the whole directory and FAT of the disc, should it become corrupted. Figure 1 shows the command in action, and the result.

Used in this way RECOVER searches the disc and assumes that each group of linked sectors is a file. It has to assure that the data in the directory is invalid and gives each file a new name such as FILE0001.REC etc. Although this will cope with the problem of a bad sector in the directory, RECOVER has not looked for or repaired bad sectors in the bulk of the disc.

There are now two problems: firstly which file is which, and secondly which are the bad files. To find the bad files we must use the file-by-file form of RECOVER we looked at first. Finding the corrupted file is often painstaking, but can be helped by comparing the file lengths in the recovered directory with the original file lengths. However, you must be aware that the new file lengths will be rounded

up to whole sector sizes and will not necessarily be in the original order. When you think you have found the file, try TYPEing it to confirm you have the correct file.

CHKDSK

The DOS manual say that CHKDSK checks and repairs errors on discs, which is a little misleading. The only errors CHKDSK repairs are logical errors, and not physical errors such as bad sectors. So, CHKDSK will sort out the problem we had above where the directory entry was inaccurate but will not find, or map out, bad sectors.

Figure 2 shows the dialogue when CHKDSK is used immediately after recovering LOG.TXT which, if you remember, left us with inappropriate information in the directory. You can see that the bad sector is now recognised and the size of LOG.TXT is adjusted, or is it..? As ever with DOS, things may not be as they seem. CHKDSK will not actually make the corrections to the directory unless you specifically ask it to. The command in this case should have been:

```
CHKDSK B: /F
```

where the /F means fix it .

Ideally CHKDSK should be run from time to time to pick up lost clusters and release them for general use. Lost clusters can arise from files disrupted by bad blocks as we saw above but can also arise if applications are not shut down properly, for example, by a sudden power failure or computers being turned off unexpectedly.

BACKUP

Basic data security can only be maintained by a proper backup schedule. We all know this and we all cheat (at least, until we lose a hard disc full of data.)! The conventional advice for a hard disc would be as follows:

1) Do a complete backup to floppies at the same time every month.
2) Do an incremental backup (that is only those files which have been changed since the last backup) at the end of the working day onto a second set of discs.
3) At the end of the next day do the incremental backup onto a third set of discs.
4) Alternate the second and third set of backup discs from day to day.

To do a complete hard disc backup the command is quite simply:

```
BACKUP C:\ A: /S
```

The /S switch makes sure that all sub-directories are backed-up up as well. All you have to do now is to insert floppy discs when told to. You will, in all probability, need a large number of floppy discs and the process will take a long time. You can append an /F switch to the backup command to tell the program to format the floppy discs prior to storing the backup files on them. I prefer to pre-format them using the ADFS format because this enables me to reject any with bad sectors. To estimate the number of discs you need, take the total disc capacity for your hard disc, subtract the number of free bytes (given by DIR), divide by 1000, and then by 720 (for 720 K discs) and then add one or two for good measure.

Incremental backups are performed using the command:

```
BACKUP C:\ A: /M
```

where /M means modified files only. All DOS files have an archive attribute bit which is set whenever a file is written to. BACKUP resets the bit after backing up the file and can tell, therefore, which files have been modified. As with all backup facilities you can also archive only those files modified after a certain date. Another feature I find useful is the backup log. For example:

```
BACKUP C:\ A: /S /D:13-03-93 /L:BACK.LOG
```

will backup all files on the hard disc created on or after the date given, and leave a list of all the files together with the number of the backup disc in a file called BACK.LOG. Afterwards this can be sorted using:

```
SORT /+6 < BACK.LOG >
```