The upper hand

Locating resident programs and drivers in Upper memory, speeding up indexing and sprucing up MSDOS 5, all calmly dealt with by Simon Collin.

Several issues ago you published an ‘ideal’ configuration for a standard PC with 4Mb of RAM. I followed this and have freed up some extra RAM, but I’m not too sure which of the resident programs and drivers are loaded where. To make the most of my setup, I would like to know if there is any way of telling where a program is located. Is it in upper memory, for example, since I don’t think my configuration is making the best use of upper memory.”

PCW *Upper memory is not difficult to deal with and is a very useful area in which to store resident drivers out of conventional memory. The upper memory area lies between the 640Kb and the 1Mb mark and is supposedly reserved for use by hardware. A video card or network adaptor will have a fixed address in upper memory that transfers data to and from the processor. Since these devices use a fixed address, you need a special memory manager that spots if an address is in use and which range is unused.*

A memory manager, such as DOS’ EMM386, will associate these UMB (upper memory blocks) with real memory locations above 1Mb. Once this is done, you can store resident programs or data in the UMBs. The first step is to check that you have told your memory manager to look after the upper memory area and support UMBs. In the case of DOS’ EMM386 you have to follow it with one of two parameters: the NOEMS switch will provide UMB but not EMS support, while the RAM switch will provide support for both UMBs and EMS memory.

I have covered how to set up UMBs and EMS in depth in previous columns, including the one to which you refer, so I’ll skip this section and move on to your question: how do you see which program is using which area of memory. To do this, you can use the standard DOS MEM command with the switch, MODULE. This will display a detailed listing showing where a particular program is stored in memory and how much memory it’s using. For example, if you have loaded SETVER high (with the LH command) you could see where it is located using the line

```
MEM /MODULE:SETVER
```

MEM displays a report with details about the program; in this case it would look something like Fig 1.

To check if a program has been loaded into upper memory, look at the Region column: if there’s a number in there, it means that the program has been loaded into upper memory. If there are several numbers under the Region column, don’t worry, it just means that the program was too large to fit into one managed UMB and has been split over several.

Up against the BUFFERS

“I use my PC for indexing book manuscripts and would like to speed up the process. The DOS program looks through each chapter file and builds an index. For many of the large scientific works there can be hundreds of sections. The culprit is probably the hard disk since its activity light is on the whole time. I have tried to adjust the BUFFERS and the FILES commands in the CONFIG.SYS file but I’m not sure how high to set each.”

PCW *BUFFERS will always cause problems, for the reason that users are never quite sure what it does. If you just increase its setting, you could easily cut performance rather than boost it. However, if you use it properly you can increase the read-write performance of your disk drives and regain a little RAM.*

DOS reads data from a disk in 512-byte chunks, representing one disk sector; this data is temporarily stored in a buffer. If the data doesn’t fill up the buffer, then data from the next file is also read in, which — with a cacheing algorithm — is used if you then request the next sequential file. In short, the buffers try and cut down the number of reads and writes to disk to speed up your programs.

A single buffer is 512 bytes long (although it actually has an extra 16 bytes tacked on for system use by DOS). Many users think that if they increase the number of buffers, so the performance will increase. This is true up to a point, but after you’ve passed the optimum setting you’re just wasting memory. The setting depends a lot on the size of the hard disk, so start with these basic values:

less than 40Mb	20
40-79Mb	30
80-119Mb	40
>120Mb	50

If you don’t have any cache software such as SMARTDRV installed, then there are more ways to improve performance using the BUFFERS command which are particularly useful when accessing sequential files. The BUFFERS command can (from MSDOS 4.0 or later) take a second parameter,

```
BUFFERS=30,4
```

The second parameter, in this case 4, defines the number of consecutive sectors that are read in each time DOS carries out a disk read. This is particularly useful when accessing lots of sequential files and can dramatically improve access times.

Since your indexing program reads a mass of sequential files, getting the BUFFERS statement set up correctly should produce good results.

Catching up with commands

“You have been concentrating on the newer features of DOS 6 and its contemporaries and are leaving me behind! I am still using an elderly version of MSDOS 5, and use batch files to spruce it up and give it a bit more zing. I would like to add that the overwrite copy protection that I have seen is part of MSDOS 6.2; is there any way of doing this with a batch file command that would work on my system?”

PCW *There is a neat way of detecting an existing file that has the same name as the source file using the IF EXIST command in a batch file. The simplest way of using the EXIST command is to write it into a batch file called CHECK.BAT. If you enter this with a filename as an argument (for example, CHECK LETTER.DOC) it will report back if there is already a file with this name.*

```
FOR %%I IN (%1) DO IF EXIST %2/%%I
ECHO %%I already exists
```

The FOR loop uses a variable called “I” (it could be any single letter) and looks at the two command-line parameters that were used with the batch file. You would use CHECK.BAT as follows:

```
CHECK LETTER.DOC \FILES
```

The line of code would check through all the files in the directory named in the second parameter (%2) for the file that is to be named in the first parameter (%1).

This line does nothing more than check if the named file already exists in the target directory: if it does exist, the batch file will display a warning message. To turn this into a copy routine that prevents overwriting takes a slightly different line of code:

```
FOR %%I IN (%1) DO IF NOT EXIST
%2/%%I COPY %%I %2
```

The complete batch file, now called COPY2.BAT, would have two lines and look like this:

```
@ECHO OFF
FOR %%I IN (%1) DO IF NOT EXIST
 %2/%%I COPY %%I %2
```

This technique is very useful, not only for updating old versions of DOS, but also to pass on processing to another program. The only snag is that it cannot manage wildcard copies, but for most applications this is not a handicap.

Fig 1 MEM report

```
SETVER is using the following memory:
        Segment  Region           Total       Type
        ───-  ───  ────────  ────-
        OE801     2               768   (1K)   Program installed:SETVER
                                ────────
        Total Size:               768   (1K)
```

Windows 95: how does it affect DOS users?

With the arrival of Windows 95, the number of letters I have received about its effects on DOS users has increased exponentially. I mentioned some of the more cosmetic changes that DOS users can expect, together with the use of the scalable TrueType font technology, in last month’s column.

As far as readers of this column go, at the heart of Windows 95 beats a new version of MSDOS. Microsoft was initially going to remove all traces of DOS from Win95, but has bowed to pressure and added what would have been MSDOS 7 to the setup. There are also rumours that Microsoft will be releasing this as a separate, standalone product, although when is not clear.

When you start up Win95 there is little sign of MSDOS unless you are loading device drivers or TSRs from the CONFIG.SYS file, in which case Win95 switches mode to load these programs, then goes back to its native mode. The DOS commands under Windows are tucked away in the \WINDOWS\COMMAND directory and include all the standard MSDOS files that you would expect. The Windows tools, such as antivirus and backup, are integrated within Win95 so no longer appear in the COMMAND directory. There are only a few enhancements to the basic feature set, with a new switch for the DIR command and, of rather more interest, a new command called START.

The DIR command has gained a “/V” switch to display the files in verbose mode. This lists one filename per line together with date, size, version, and full attribute set. In addition, the listing ends with the total disk space and the percentage that is used up.

The new START command is far more exciting, since it lets you run any DOS or Windows application from the DOS command line. Not only can you start Windows programs from DOS, but you can also define how they run: the options include minimised on startup and full-screen.

For example, if you want to run the Windows calculator from the DOS command line, just enter “START CALC” and Win95 will load and run the file. There are four option switches to START: “/m” runs the programs as a background (minimised) job; “/max” runs it full-screen in the foreground; “/r” runs it in its default mode; and “/w” will wait until the program has finished before returning to the DOS prompt.

As you can imagine, this suddenly gives a whole new range of possibilities to the humble batch file since it can now run Windows programs and process their results. In short, batch-file programming can be used as a simple scripting language to control Win95 and DOS.

PCW Contacts

Write care of *PCW* or via email to **scollin@cix.compulink.co.uk** or **CompuServe 72241,601**