RANDOM SELECTION

In the last part of our look at file handling we discussed sequential files. Now we look at an alternative but complementary technique — random access files. Although this type of file offers very direct and therefore faster access to data, it uses more storage space and must be carefully and uniformly defined.

The limitations of sequential files arise because of the necessity to read the information stored in them in order. Random or direct access files provide a solution to these limitations because the records within them can be accessed in any order and very quickly. The word 'random' does not imply that the file is constructed or used in a chaotic manner, it simply means that any segment may be written to or read from without the need to go through all the preceding information.

The obvious problem here is that all files held on cassette tape must be sequential files. There is no way to go straight to an item of data in the middle of a cassette tape; instead the whole tape must be read through. The only way to use random access files on a micro that relies on tape storage is to load all the data into memory, but this limits the file size. Disk drives are needed for useful random access; but even so, a few makes of disk drive cannot support this type of file handling.

The user will also find that random access files are easier to work with than the rather cumbersome techniques needed for sequential files. The division of the file into records and fields that we detailed on page 226 is very important with random access files. To access the file, the required record must be specified. This record, together with its fields, will then be put into a buffer in the computer's memory, where the fields may be deleted, amended or printed.

Fortunately, the operating system will take care of the more complicated structures necessary. It will need to go quickly to the start of a particular record on a disk. It cannot do this on a sequential file, as the only way to locate a record is to read through all the data, counting off each field marker. In order to facilitate rapid location, every record in a random file is the same length. If each record were 100 bytes or characters long, and the program asked for record number 83, the operating system would position the disk head at the start of the 8300th byte of the file. It has a record of how many bytes are in each sector of the disk and can therefore calculate the location of the required record.

This method of file reading may seem complicated, and it is certainly slow, but it is far quicker than reading through a sequential file.

When standardising the length of the files, it is obviously necessary to choose a size that will accommodate the longest record stored in the file. Shorter records must be padded out, usually with spaces (32 in ASCII code). This is a major drawback to random files, as the padding required to make the records up to length is a waste of precious storage space. This means that random files are used for small amounts of information where access needs to be quick, while sequential files are used for bulk storage where access speed is unimportant.

Fields within a record must also be set to a standard length. This is particularly relevant to systems that provide a random access facility to specific fields as well as to particular records. For systems without this facility, it is still a neater and more efficient way of file definition. The first step when designing a random access file is to list the different fields and decide on suitable lengths for them. A field for a person's name should be at least 20 characters long, for example, whereas you would need only two characters in which to store their age.

Economy is crucial when designing a file, as there will inevitably be a trade-off between the amount of information stored and the number of different records. Quite often, coding systems can be devised to reduce the amount of space taken up

Random Vs Sequential Files

	RANDOM ACCESS FILES	SEQUENTIAL FILES
PROS	• Fast access to particular records	• Conserve space • Available on tape systems
CONS	• Waste space • Need disks	• Slow and cumbersome
SUITABLE APPLICATIONS	• Predictable data that is in a defined format • When small numbers of different records are accessed; for example, in a library where customers ask for details of particular books. This is a low 'hit rate' application	• Large quantities of unstructured data • When most of the records in a file are processed in a single run of the program; for example, in a salary system, where every employee must be paid. This is known as a high 'hit rate'