ORDERLY PROCESSION

The sequential (or serial) file is a heritage of the days of tape-based data processing; if magnetic disks had been the cheapest form of mass storage for the last 40 years, then our methods of handling data would be very different indeed. In this article we discover how to create and access sequential files from a BASIC program.

The binary (or program) file is a simplified case of a sequential file. When you type SAVE"progname", the operating system performs the several discrete operations required in creating a file: first, it opens communication with the tape or disk drive, and writes a header label containing the file name and some information about the file contents. Next, the operating system writes the block of memory containing the current program into the file. Lastly, it writes an end-of-file marker to the file, and closes communication between the computer and the tape or disk drive.

The BASIC commands used in creating a sequential file vary from computer to computer, but they must all carry out the same tasks. Taking the Commodore 64 as an example:

```
100 R$="This is one record of the file"
150 OPEN 5,1,2, "DATAFILE,SEQ,WRITE"
200 PRINT#5,R$
250 CLOSE 5
```

This starts with the OPEN 5,1,2 command, which prompts the operating system to establish a channel of communication, called channel 5, to device number 1 — the tape drive. The operating system can create sequential files on a number of different devices, and it can access several different files at one time, so the channel from computer to device and thence to a particular file must be labelled. (The number 2 is specific to the Commodore and unimportant for the moment.)

After the OPEN command comes the file name. On the Commodore the file name consists of a name such as 'DATAFILE', followed by the file type (SEQ, standing for sequential file) and the access method (WRITE indicating that the file is being OPENed for writing). Sequential files can be OPENed for either writing to or reading from, not both at the same time. The effect of this command is that the device read/write head is energised, and a 'header' record consisting of the file name and some system information is written to mark the start of the file.

The PRINT#5 command in line 200 sends data to the file. PRINT has its usual meaning, but the '#' sign indicates that data is to be sent to the specified channel rather than to the screen — the default output channel. The contents of R$ are, therefore, sent down channel 5 to the sequential file called 'DATAFILE'. The file now contains one record — the string, "This is one record of the file". Finally, CLOSE 5 closes channel 5 to further communication, and writes an end-of-file marker onto the file.

Information in a file is intended to be read at a later stage, for example:

```
500 OPEN 3,1,2, "DATAFILE,SEQ,READ"
550 INPUT#3, A$
600 CLOSE 3
650 PRINT A$
```

Both this fragment of program and the previous one use the OPEN command, here meaning 'find the beginning of the file called DATAFILE and prepare to read from it', whereas previously it meant 'create a file called DATAFILE, and prepare to write to it'. The channel numbers are different, but they are simply labels (a different number could have been chosen in both cases without affecting the operation); the device address, however, is the same in both cases because the file is stored on device 1 — the tape drive. The file name and the file type are the same because they identify the file to be accessed, but the access method is different — READ instead of WRITE.

This program has INPUT#3,A$, meaning 'input from channel 3' instead of from the keyboard — the default input channel. The first complete record in the file is read and sent along channel 3 into the variable A$, just as it was sent along channel 5 from the variable R$ by PRINT#5,R$.

Some of the limitations and most of the advantages of sequential files are thus revealed: they can store anything that a variable can hold, and there are no restrictions on file size or structure. On the other hand, the contents of a sequential file must be read from the start of the file, record by record; there is no way to open the file at a particular record, nor to skip, re-read, delete, insert or append a record.

BOB HALL

Order Of Play

Sequential files were developed with tape storage in mind; but the order in which records are sent to the file is the order in which they must remain. The only way to sort or edit a sequential file is by creating a new version of it elsewhere on tape, just as a music cassette can be edited only be re-recording or by physically cutting the tape