FILES

WHAT ARE FILES ?

If we pursue our top-down development rigorously, then files are simply the things we wanted as long-term repositories for our "property" in the computer system. We therefore start with the idea that files are collections of any sort of information we might want to save; and, as we'll want to find them again, they must have names of some sort. We don't, at the moment, care much what sort of machine they live in; we hope that, by studying what we want to do with files, we'll be able to specify them more closely, and then decide how to make them work.

That isn't how files developed. They grew up in the same scruffy and haphazard way as most other computing things. Because of that, one difficulty in talking about files is that no one (except, of course, us) is quite sure what they are. In the early programming languages, a file was simply something you manipulated with a **read** or **write** instruction – which is to say, any useful source of, or destination for, information which is not directly addressable by the programme. The most obvious difference between the two "definitions" is that the traditional version includes input from and output to terminals and similar devices. In a later chapter (*FILES – FROM THE BEGINNING*), we offer a homily about some of the things called files, with particular reference to their permanence or otherwise.

So far as implementation was concerned, the early developers did not follow our Olympian policy of first working out how files would be used, and then deciding how to do it. All they knew about files was the basic property – we need some sort of long-term storage device – and they used any technological means available to do that, which in practice meant paper tapes and punched cards, soon superseded by magnetic tapes and (later) discs. (Before computers came along, many commercial organisations had established data processing departments based on punched card systems, for which much very efficient specialised machinery was available. In that case, though, the punched cards were both the storage medium and the processing medium; they served the same purpose as the modern internal memory as well as being conveniently permanent. With the much faster electronic processors, card handling was far too slow to be useful for filing purposes.)

In this chapter, and indeed throughout our discussion, we stick to something like the long-term data repository "definition" according to top-down analysis, and contemplate some immediate implications of that view. The general nature of the contemplations will be in some respects familiar to you, for we are after all discussing data structures, and we seek the most appropriate structure for data storage. In this context, we must go beyond the usual abstract view of data structures, and think about issues such as how to store them and how to find them when we want them, which are necessary for files in the operating system, but even these topics have their parallels in implementation and use of data structures. The significant new feature – and the reason for our new concerns about storage and identification – is the requirement that the files should be *persistent*, outliving the processes in which they are used, so we have to build independent structures within which we can organise the files.

What must the operating system know about a file ? We can easily list all the sorts of thing there are to know about a file :

- The **name** : If the file is a recognisable entity within the system, it must be identifiable in some way, so there must be something which functions as its name. This may be a text string, or it may be an icon, or it may be an address on the storage medium. Whatever form the name takes, it can be used in principle to find the file. ("In principle", because it might be that information about the file is not immediately accessible to the system (the file is on a disc which isn't mounted anywhere) or the file contents might not be accessible (they are in the archive).)

- The **properties** : There are many facts we can know about files, some of which are more useful than others. We might know a file's owner, time of last use,

security classification, size, etc.; all these items might be of use in some circumstances, but none gives any direct information about the file's contents.

- The **contents** : The material which forms the body of the file is clearly known to the operating system in the sense that all its details are accessible to the system, but it is usually supposed that the contents are not the system's concern.

In an implementation, we must provide storage for the contents of the file, but there is room for flexibility in choosing how to handle the other items. You can even ignore them entirely so far as the hardware and software are concerned; if each file is recorded on a separate tape or disc, then the name can be handled purely clerically (by writing it on the tape label). Some of the properties must be known in order to write a programme which can read the file and make sense of it, but these too can also be left as tasks for the implementers to handle independently of the computer system. In practice, this extreme position is unworkable. We want to be able to store more than one file on a disc, so we must have some provision for identifying the file we want and finding it; and if we want the system to do intelligent things with the files (such as protecting them from unauthorised access, making backup copies, etc.) it must have access to such information about the files as it needs to perform these operations. We therefore see that a useful file store is likely to provide access to the contents and (at least some) properties of files identified by name; we have already started to design the system !

NOTICE !

We insist that in this section we are discussing files in the abstract, without any commitment to any specific storage medium. Despite this, if we are to present any examples at all we shall have to mention a storage medium – and this is bound to be disc, as the vast majority of computer files live on discs of one sort or another. Please remember that this is an accident, not in any sense an essential property of the files.

QUESTIONS.

We said, "With the much faster electronic processors, card handling was far too slow to be useful for filing purposes". In the earliest systems, the computers did the same job as the card systems they replaced – so why was card handling too slow ?
