FILES AND STREAMS – IMPLICATIONS

FILES.

It seems that we have decided that we need some provision for storing collections of information, which we shall call files, in our computer system; we must therefore ask what facilities the system should provide in support. We begin with the basic properties of a file :

- It requires storage space; and
- It must be accessible to programmes.

We therefore need :

- Somewhere to put them.
- Ways to find them.
- Means to get the information in and out – **read**, **write**, etc.

Even that apparently simple statement needs qualification. We've argued that some files aren't really files at all – they're streams, so they should be handled quite separately – but this isn't a widespread view. Nevertheless, it remains true that some files are ephemeral – typically files associated with terminals – and don't need any permanent storage space; and demands of security might mean that some files must only be accessible under certain conditions.

USING THE FILES.

To make the files work with programmes, the system must provide support both as means of performing operations and as data. The programmes must be able to do things to the files, and to do so they, and the system, must have information about the files which are to be used.

System procedures : The system must provide software which ordinary programmes can use to communicate with each file, typically as procedures protected through the supervisor call mechanism. The facilities provided must be complete, in the sense that any operation of the file, or information about the file, which someone might reasonably want to use should be available.

File tables : We must be able to defer binding a file identifier to a specific file for as long as we can if device equivalence is to work properly. This means that a programme must be compiled in terms of abstract files, and the code file must include enough information for these abstract files to be associated with real devices or other procedures (as in pipes) when the programmes are executed.

STREAMS AND DEVICES.

The file vs. stream difference appears once again as we think about what to do with the collections of data, whatever they are. The equivalent to "Somewhere to put them" for streams is something like "Somewhere to send them". We have thought of streams as terminals, but they are not the only examples : printers, communications lines, tape drives, document readers of various sorts also either are, or can behave as, streams.

As that list illustrates, a common characteristic of a stream is its association with some hardware device, and this can lead to rationing problems. Nowadays, it is usual for everyone using a computer system to be provided with a terminal; but other devices might not be so lavishly supplied, and there are then questions of access. What happens if two or more people want to send a file to a printer at the same time ? It is obviously silly to give both access simultaneously – so we must have :

- some sort of temporary storage mechanism for streams.

This is commonly called spooling, for historical reasons (see the chapter *MEMORY TRICKS*). It is also still useful for its original purpose – to compensate for the speed difference between the programme and the printer.

What do we mean by "temporary storage" ? Among other things, we mean we have a collection of information we want to store for some time. We know all about that : it's a file !

PROTECTION AND SECURITY.

For archiving :

- some suitable archive device, and software to drive it.
- provision in the normal system to select files for archive.

For file protection :

- The *security requirements* don't change the basic properties; but they must be kept in mind during the implementation. You don't just open a file on request; you check first to see who's asking. This is exceedingly untidy, as in itself it has nothing whatever to do with the file system, but life is like that.
