FILE PROTECTION

Computer files stored on tape or disk can generally be accessed by anyone who knows the name of a particular file or can get a catalogue of the files available. If a file contains confidential information (a company's accounts, for example, or the source code of a commercial program) it must be protected from unauthorised users. The simplest method involves locking the disk or tape away in a secure location, and some companies spend fortunes on this kind of security for their computer systems, recognising that the information contained is their most precious resource. Physical protection, however, is compromised by the need for people to use the system, and disks, tapes and security passes can go astray. Against these failures, therefore, the creator of a confidential file has to provide some form of built-in *file protection.*

A large multi-user system may provide the first level of protection by requiring users to *log-on* with an authorised user code and a user-defined password. The user code (sometimes called a PPN — programmer project number) determines the user's *system privileges,* including the level of access permitted. Each user has an individual file directory area on disk, and the proper combination of privilege and password is required to access other users' directories. System personnel, such as operators and programmers, often have common PPNs and passwords — like a hotel's passkeys — that can give access to the entire system. Because of the power this gives, and because the codes are often unimaginatively chosen, cracking this protection is a hacker's prime objective (see page 486).

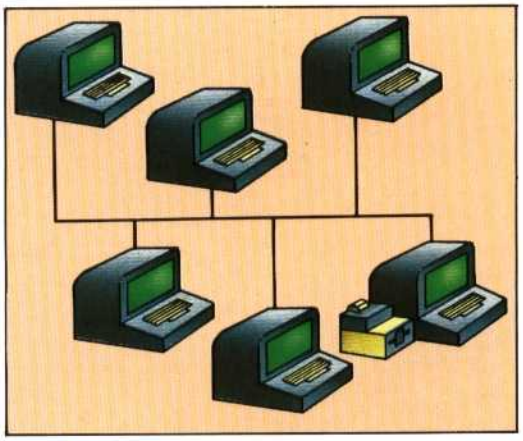
A more widespread problem than that of breaking into private data files is *software piracy* — the illegal copying of commercial program files. Software publishers have tried many forms of file protection, only to find most of them being broken as soon as they are introduced. The earliest file protection techniques included hiding data in normally unused spaces on a disk or tape. This method confuses the computer's operating system so that it can read the file only when the program is running, and cannot copy it. A more recent approach to this problem involves serialising disk-based software. The first time a program is run on a computer, a serial number associated with that particular computer is added to the file. From then on, that file — and any copies made from it — will work on that machine only.

FILE SERVER

In a network, one of the computers is sometimes set aside to handle file accesses for all the other terminals on the network. In such a system, all communications from and to the terminals, printers, disk drives and other peripherals are treated as separate files, which greatly simplifies the controlling computer's task. This computer is called a *file server,* and its only task is to control the movement of files between nodes on the network and the storage unit.

The file server is called by a node when a user needs a particular file (which could be a data or program file on disk, or a communication with a printer or other peripheral). If the file is open to that user and not being used by another node, the file server sends the data. The file is *massaged,* or altered, by the user and returned to the file server, where it is updated to incorporate any changes and then re-saved.

The Econet system is a low-cost and popular example of this kind of network, and normally it allows a BBC Micro as the file server. Users of the early versions of the system, however, found that more processor power was needed to run a network of any size. The Acorn System 3 was generally used as a stop-gap until the 6502 second processor for the BBC became available.



Joining Forces
The Acorn Econet joins up BBC Micros in a bus-type network. The bus may be a simple twisted-wire pair, and one micro is dedicated as the file server

FILE TRANSFER

A file is usually created within a computer and then stored in some peripheral storage device. Copying the file between them — or to another node in a network, or to some distant system — are the commonest examples of *file transfer.* The problem in these vital communications is the recurring one of format compatibility. All communicating devices have their own requirements (or *protocol*) concerning the transmission rate, parity, and connect/disconnect signals. These protocols are often specific to the device, so file transfer is rarely a simple matter.

Different manufacturers and organisations have made various attempts at standardising these formats, but their solutions have usually added to the problems. The recent growth in national and international telephone data networks, however, is gradually imposing consistency. Within a network, the simplest answer is usually to make all transfers through the network controller (file server), which can accept a file from any node and pass it to any other node in the network in the appropriate format.