ARCHIVING

Once the file system is running, it begins to fill up with people's files. So far as is known, this process is irreversible (short of a nasty accident to the storage medium); everyone accumulates files which might always come in useful some day, and few manage to throw away any but the most obviously temporary files. In consequence, whatever medium the system uses as its file store soon becomes full, with more or less alarming consequences.

Aside from the difficulties caused by the sheer volume of data which people wish to preserve, there is the question of safety. Even though machinery and software are far more reliable now than they used to be, breakdowns do occasionally happen, and security breaches are not unknown; as well as that, people have never been very reliable, and continue to make mistakes. There is therefore a demand for secure off-line storage where valuable information can be safely kept until needed without risk of loss or unauthorised access. System backup is not the answer, for, even if a good copy of a file is stored on a backup tape, it might be quite difficult to find just the version of the file which you want if you are not quite sure precisely when the damage occurred. In any case, backup happens when it's convenient for the system, and there's no reason to expect that it will happen when it's convenient for you. One normally wishes to save a copy of a file each time it has reached some stage of development – the next version of a working programme, or the cumulative file updated to the end of each month. This can only be guaranteed if the storage is under your control.

Both these defects in the file system are addressed by *archive systems*.

AIMS AND PRINCIPLES.

An archive system is intended to provide additional off-line and possibly long-term storage for files, to increase the system safety, and to reduce the demand for on-line storage. The archive is typically kept on one or several magnetic tapes, so files in archive remain unaffected by system failures. It is also important that the archive remain unaffected by medium failures, so at least two copies of the archived files should be kept, on different tapes.

CLASSIFICATION.

Archive systems might be *automatic* or *discretionary*.

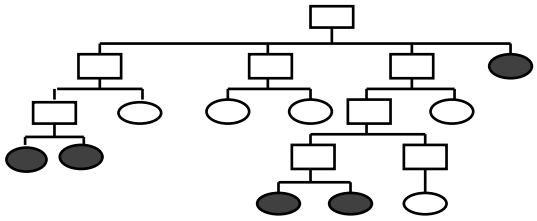
In an *automatic* archive system, the emphasis is on convenience with protection against loss or damage as a sort of by-product. The archive acts as an extension of your normal file store, and a file in the archive appears in the file table in just the same way as any other file. Files are moved into archive by the system according to some criterion such as not having been used for a set time. The primary intention of an automatic archiving system is to avoid disc congestion, so files really are *moved* into the archive and deleted from the usual file store.

Whether or not this automatic archive really qualifies as an archive system is debatable. It does release you from the constraints of a finite disc (though that isn't as important as it used to be); it doesn't give you the protection of a safe copy of a file you're using, as only one copy is maintained. You can make it work as a real archive if you want to (for example, by making a copy of any file you wish to preserve, putting it in a directory called "archive", and leaving it there), but it isn't really designed for that purpose. A good alternative view is to think of the automatic archive as just a way to implement the abstraction of an infinite disc – in very much the same way as virtual memory is a way of implementing a greatly expanded memory. We rather incline to the "virtual

disc" view, but we've kept the discussion in this section so that we can more easily compare the two techniques.

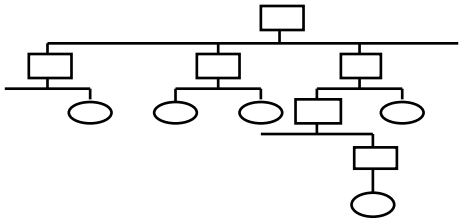
In a *discretionary* archive system, the emphasis is on safety. Files are *copied* into the archive on request, and they are copied back when required. Once in the archive, therefore, a file will never be removed, unless an explicit **delete** operation is provided. The archive is independent of the normal file system, and archiving does not imply removal from the file store. Of course, once a file is archived, the owner might choose to remove it from the file store; so files sent for archive must be immediately copied by the archive to a reserved area, and later copied to the archive tapes from there.

The difference between the methods is illustrated by the diagrams below. First, here is a diagram of a collection of files, some of which – the shaded ellipses – are in archive. This diagram applies directly to the automatic archive; all the files, archived or not, are still held in the one file system, and only files (not directories) are archived. (The directories must remain on line, for it must always be possible to make enquiries about all parts of the file system.)

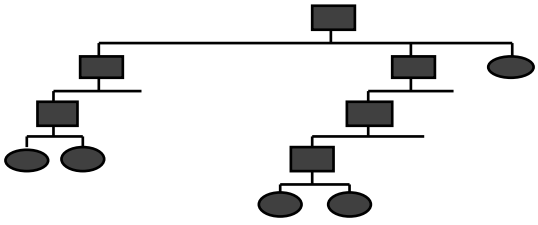


In contrast, here is a diagram of the same set of files with the same subset archived as it might be found with a discretionary system :

On disc :



In archive :



There are two significant differences : the archived files, and even some archived directories, do not appear in the ordinary directory; and a directory structure is maintained in the archive. Both of these are reflections of the fact that there are two parallel, but essentially separate, file systems. Because of this it is quite usual to have copies of certain files both on line and in the archive, though in order to emphasise the correspondence between the two systems no such case is illustrated in the diagram.

OPERATIONS ON ARCHIVED FILES.

What should you be able to do with your archived files ? There are two sorts of operation.

- First, the archive is a file system – so it is reasonable to expect that at least some of the operations which can be performed on files in the ordinary file system will also be applicable to files in archive. While it is clearly *not* reasonable to expect to be able to do things like reading and writing to the body of the file, we saw earlier that

there were several operations which applied to the file as a whole or to its attributes, and could be performed to a considerable extent using only the information in the system's file tables. If we wish to make these operations available, it's comparatively easy, because the information is already there in the ordinary file tables for an automatic system, and we need only preserve the attributes in a file table for a discretionary system.

It is certainly useful to be able to **delete** a file from archive, though the **copy** instruction, interpreted as an action within the archive, is not obviously useful. The **rename** instruction is also of doubtful value – and it would be quite hard to implement, as the operation would not be complete until all copies of the file (and there might be several of different generations, all stored at least twice) had been treated. Some instructions are connected with security : archived files are usually only accessible to their owners, but it might be convenient to make them accessible, permanently or temporarily, to other people, and instructions to effect these changes might be provided. Protective measures might also be available. Many archive system permit several versions (generations) of a file to be saved, and there might be instructions analogous to **restore** (See the chapter *FILES, STREAMS, AND PROTECTION*) to manage these. Other instructions are perhaps a little less obvious. For example, an expiry date is usually associated with an archived file; to extend the life of the archived file, there must be operations which change the expiry date.

Though we can make the operations possible at any time, that isn't the end of the story. The operations are not complete until they have been implemented in the archived material itself – an interesting property of an archive system is that its operations can typically only be effected when the archive tape (or whatever) is accessible to the system, so requests for the operations must be saved until then. This constraint has a number of interesting effects : for example, a request to delete an archived file does not result in the immediate removal of an entry from the file table, though the entry might no longer be listed; instead, the table entry is kept, but flagged as a delete request, until all copies of the deleted file have been removed from the archive. The system should keep permanently available as much information as it needs to check that instructions are properly formed, but the operations themselves must be deferred until they can be executed. The permanently available information should include at least a list of all archived files, and the names of the tapes on which copies of the files may be found.

- The second set of operations is associated with transferring files between the normal file system and the archive. The obvious pair are **archive** and **retrieve**, which copy a file between archive and the main file store. Notice that **archive** is still a useful instruction in an automatic archive system; if you know you won't want a file for a long time, it's sensible to consign it forthwith to the archive and thereby release file store space for more urgent work.

- The third of the two sets of operations is included here only for emphasis, because it can really be included in the first set. This is the set of operations with which a file's owner may get information about the file – the archive equivalent of access to the file table and information contained therein, as commonly displayed in lists of file directories. We would like at least to be able to inspect the name, date of archive, and expiry date of each file in archive. We have already noticed that the system must keep available a certain amount of information about the archived files, and the information we wish to display must be included too, if it isn't there already.

One additional item, not often provided by ordinary file systems, is convenient. If you forget what is in an accessible file, it is usually easy to inspect it; but you can't do that with an archived file, so, unless your system allows unusually expressive file names, it's helpful if the information on an archived file can include a comment of reasonable length which you can use to describe the file.

RUNNING AN ARCHIVE SYSTEM.

The real work of archiving is a batch operation, traditionally performed at night when the machines are comparatively lightly loaded.

At the beginning of the run, the system works out what it has to do. For an automatic system, the on-line file tables must be inspected and files which satisfy the criteria for archiving (typically those which have remained unused for some set time) identified. Requests for retrieval and deletion have been recorded throughout the day. An **archive** request might also require that an old version of a file already stored in the archive be deleted. For a discretionary system, all requests are explicit and have been recorded, so no directory inspection is needed. The system can then inspect its tables to identify a subset of its archive tapes which must be read to retrieve the files which have been requested; this is the minimum work which must be done. Requests for deletions can usually be deferred until the tapes carrying the copies happen to be needed provided that the deletions, together with expiry date changes, and other operations not requiring access to the file body, are recorded in the archive directories.

The required tapes are then mounted and copied to new tapes, with appropriate modifications : files to be retrieved are copied back to the disc (and, in an automatic system, not copied to the new tape); files to be deleted are not copied to the new tape; new files for archive are written (for a discretionary system, from the archive storage area) to the new tape. In addition, it is sensible to copy the complete archive directory to every archive tape as protection against its accidental loss or corruption.

As with any file system, the question of identifying the stored files must be addressed. At first sight, this seems trivial – the files already have names, so why not use them ? On further inspection, this turns out to be an oversimplification. It is just about workable with an automatic archive system, though even here care is needed. The general problem is that names embedded in an archive tape are stuck there until the tape is copied, and you don't want to copy tapes more than you need because it takes time. If you use the simple file name – the last component of the pathname – then there is a high probability of duplication. If instead you use the complete pathname, what do you do if the directory structure is changed between storing the file in archive and retrieving it ? Consider too that there might be several generations of a file in the archive, all with the same name, and all duplicated for safety. (The problem of multiple generations might arise in automatic as well as in discretionary systems. When a file is retrieved, it might be that only one of the several archived copies is physically deleted from the tapes, and the others could remain for some time unless special tape operations are inserted to get rid of the redundant material.) Many systems therefore use synthesised names within the archive file system, linking these with the files' original names only through the archive directories, which are comparatively easily accessible.

An interesting feature of an archive system depending on magnetic tape – which is still common – follows from the aggressively serial nature of the tape medium. In the interests of easy access, it is sensible to write the archive directory at the beginning of the tape, so it has to be written first. The whole operation must therefore be completely planned before any tape operations are started, and the plans must be guaranteed to work out. If anything goes wrong part way through the operation, it might have to be abandoned and restarted from scratch.

LINKS WITH THE OPERATING SYSTEM.

A discretionary archive system can be added to an operating system with little mutual interaction. It must be possible to add a programme to the available software in such a way that it can be executed by using a corresponding new instruction, similarly added to the system repertoire; this is straightforward on (we think) all common operating systems. It must be possible to reserve an area of the file store for temporary storage of files moving into and out of the archive, and to move files between the reserved area and the normal file store; this is also commonly possible. A rather more taxing requirement is that the archive software must be able to move files to and fro as described, but no other access to the reserved area should be permitted.

It is clearly an advantage if the archive system and the ordinary file store system can be integrated, so that full lists of file holdings can be made if required, and information

about the archive displayed in the ordinary file lists (for example, the date of the most recent archive can be helpful), but integration is not essential.

In comparison, an automatic archive system is by definition integrated with the file store system, and this integration must be reflected in the file tables maintained by the system, and in the software which drives it. The file lists displayed must include archived files in the positions which they occupied before being removed to the archive, but marked in some way to convey their archived state. There are also implications for the system's file handling software as used in programmes : for example, procedures for manipulating files must at least be able to respond "file in archive" as well as "no such file". This information must be returned to the procedure's caller as a result code of some sort, for only the caller is in a position to determine the appropriate response.

QUESTIONS.

What is the difference between archiving and backup ?

In a system providing automatic archiving, what is the "right" thing to do if a programme attempts to use a file but finds that it is in archive ? Is there any satisfactory automatic response which could be built into the system ?

As costs of storage media decrease, it becomes sensible to think about preserving backups for ever. In an incremental backup system in which a complete backup copy is taken each month, increments are saved each day, and the complete backups saved for many years. Is a file archive necessary ? If no archive were provided, what sort of index to the backup system would you need ?

Suppose that a file generation system based on fragments of files is implemented in a system using incremental backup. How should the backup system cope with the generations ?

If good document readers and picture scanners were readily available, would you be less reluctant to delete old files from the disc ? How good would they have to be ?
