Integrity of a mass storage filing system

By A. G. Fraser*

The integrity of a mass storage file system can substantially exceed that of the hardware and software involved. This effect has been obtained at Cambridge University where such a file system forms an essential part of a multi-access system. A fixed disc store is backed up with magnetic tape and this configuration forms the basis of a design study which seeks to identify the problems and principles involved in file management. The result is simple but effective and involves a two tier file dumping system.

(First received April 1968 and in revised form June 1968)

A filing system is rather like a ship: if loosely constructed when launched, it will be in danger of sinking. Repairs to the structure will only partly solve the problem and bailing will certainly be necessary. The cost and urgency of the bailing operation will depend directly upon the design standard, integrity of the construction and the roughness of the sea.

A file system must be designed to detect and repel errors. Unlike water, errors can be self propagating and this unfortunate fact is particularly evident when randomly accessible bulk storage is used. It is of paramount importance that a filing system should be designed to minimise the direct and consequential costs of erroneous performance by hardware, software, users' programs, and computer operators. It is a design objective that must be faced at an early stage of software planning since it will not usually be possible to superimpose satisfactory recovery procedures on software that has been designed without regard to this problem.

The Cambridge filing system

The filing system which is in operation on the Titan computer at Cambridge University and which supports a combined multi-access and job-shop workload, provides storage for data over indefinitely long periods of time. A 64 million character disc is augmented by the use of magnetic tape. The organisation of this available space is entirely a system responsibility and the user can normally expect data to be transferred into his immediate access store on demand.

A combination of error detection and recovery techniques is used to maintain file integrity in the face of unreliable hardware and the perturbations produced by programmers developing new software. The design has been balanced to give an acceptable cost/effectiveness ratio in an environment where stoppages are frequent (several per day) and files are typically small (50% of all files each contain less than 4K characters). File dumping and the other precautionary measures that contribute to successful recovery after error, are functions that do not rely on user initiative but are effected as a matter of routine by the operating system. Recovery procedures are also initiated automatically when the occasion demands and usually follow one of the consistency checks that the system regularly applies to itself.

Massive dump and recovery

The simplest technique for recovery from error is to copy periodically the entire content of the disc on to magnetic tape (usually in duplicate) and to copy it al back again in the event of a failure. However, the time taken to copy all the information may be of the order of several hours, and this has three consequences:

1. In the event of a breakdown, it will take several hours to put the filing system on the air. Continuous service is therefore an impossibility.
2. It will not be economically acceptable to perform the dump more frequently than (say) once a week, with the result that up to a week's work will be lost every time the disc fails.
3. One will be tempted to dump the information in the sequence in which it can be most rapidly obtained from the disc and not in the logical order. This will probably mean that recovery of isolated files from old dump tapes is no simple matter.

These difficulties are mitigated if the disc is split into several conceptually independent areas, and the dump/recovery process is applied independently to each area. The load can also be made more palatable if the dumping process is carried out in frequent short stages which are time-shared with the normal computing load. However, this system does not make frequent dumping an economic proposition and the amount of lost work that follows from each failure could still be large.

The incremental dump

To minimise the risk of loss, fresh data should be recorded on magnetic tape as soon as possible after it is created, and any changes to existing data should be dumped with minimum delay. For frequent dumping to be acceptable, the amount of work done on each occasion must be kept to a minimum, even when this is done at the expense of a more involved recovery process. It is this reasoning that leads to the incremental dump program which records all changes, additions and deletions on magnetic tape as soon as possible after they have been effected.

The major drawback of an incremental dump system is that it does not yield a compact record of the most up-to-date set of data. To recover the latest data set one must scan through the entire output and determine the cumulative effect of all the changes recorded since dumping first occurred. It is therefore more realistic to combine incremental and full dumping so that the recovery procedure needs to scan only those changes recorded since the most recent full dump. A further simplification can be obtained by choosing to dump a complete new copy of a file even though only a small

* *University Mathematical Laboratory, Corn Exchange St., Cambridge*

change may have been made. This action will increase the amount of work done by the incremental dumper, but it will substantially simplify the recovery process; it is this procedure which has been adopted at Cambridge.

File dumping at Cambridge

Two dump programs are used in parallel. The primary system (**Fig. 1**) makes incremental dumps on magnetic tape at 20 minute intervals and is used to protect files until they have been handled by the secondary system. The secondary system (**Fig. 2**) maintains a compact record of all files but there is usually a delay of up to one week before any particular file finds its way into this system. A conventional tape-to-tape updating process is used in which new and recently altered files are copied from disc and unwanted files are ignored. The two systems complement one-another well, to give many of the advantages and few of the disadvantages of the separate basic methods for file dumping. Both systems are time-shared with, and take turn among, the day's normal workload so that the effective cost is kept to a minimum.

The focal point of these systems, and indeed, the basis of the entire file management system, is a directory of all known files. This directory is held on disc and is updated whenever a file is created, altered or dumped. By scanning the directory it is possible to identify files in three distinct states:

State A Files that have been created or altered but for which there is no up-to-date dumped copy.

State B Files that have an up-to-date dumped copy in the primary system but which have no up-to-date copy in the secondary system.

State C Files that have an up-to-date dumped copy in the secondary system.

Whenever a file dump is made that copy is given a unique identity. From this identity it is possible to determine on which magnetic tape(s) that particular copy of the file is to be found and it is this information that is used during the file recovery process.

The primary system

The primary dump system has its own stock of magnetic tapes which, for convenience, can be identified as *P*1, *P*2 . . . *Pn*. At any point in time one of these tapes will be in use by the primary dumper and files in state A will be copied onto it at regular intervals. When one tape is full the next is used and when all tapes have been filled the system proceeds to re-use them starting with *P*1. (See **Fig. 1**.)

There is one danger in this procedure: by using a tape for a second time we may over-write the only up-to-date copy of a file that has remained in state B for an unusually long time. In practice we try to adjust the frequency of dumping and the number of tapes used so that this event is not likely to occur. Then, to avoid any chance of accident, we arrange that the primary dump program makes a new dump of any file in state B that has its copy on a tape that is shortly to be re-used. In this way the primary system cannot be held up by some delay in the secondary system; the only effect of a stoppage in the secondary system is an increase in the volume of data that has to be re-dumped by the primary system. The need to adhere to a rigid time-table for secondary dumping has thereby been avoided.

FILES COPIED FROM DISC AT 20 MINUTE INTERVALS:

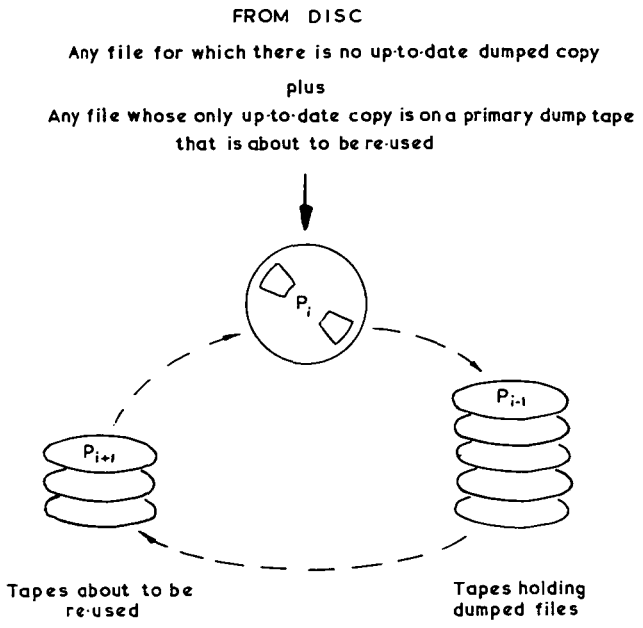

Fig. 1. The primary dump system

FILES COPIED FROM DISC APPROXIMATELY ONCE PER WEEK

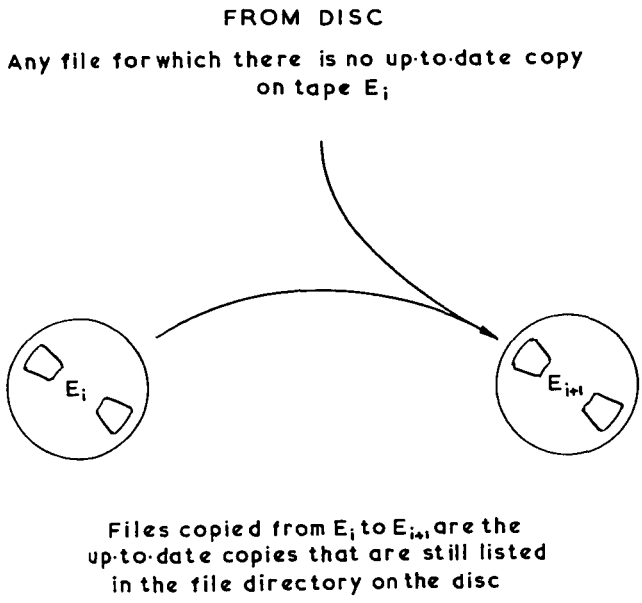

Fig. 2. The secondary dump system

The secondary system

The secondary dump system also has its own stock of magnetic tapes. These are grouped into a number of distinct sets of tapes and each set is identified by a unique letter. The tapes of set *E*, for example, will be identified as *E*1, *E*2, . . . etc.

The population of file owners is subdivided into major groups so that the total volume of data belonging to one major group is just less than the capacity of one magnetic tape. Each major group is assigned to one set of secondary dump tapes so that the files of major group E will be copied by the secondary system into tapes in set E. At any instant, one tape, $E3$, say, will hold the most up-to-date secondary dump copies of all files owned by members of this group. When the secondary dump system next operates for this group, it will write onto tape $E4$, copying files in states A and B from disc and transferring the remainder from tape $E3$. Deleted files are ignored during this process. At any time, therefore, all the files for one user group are held on one magnetic tape. This makes rapid recovery of users possible and simplifies the search for lost files.

The secondary dump system does not work to a strict time-scale, but the frequency of dumping is determined by demand. However, to avoid conjestion in the primary system, a secondary dump task is automatically scheduled for any set of tapes that has not been updated within a fixed period (about one week).

One important advantage of this system over the conventional massive dump is the fact that it gives archive storage for files not in active use and it does so with negligible addition to the software. Since the secondary dump program is a tape-to-tape update, the system can handle a volume of data that substantially exceeds the available disc capacity. We therefore chose to distinguish between two classes of file.

ARCHIVE — This class includes files that are not likely to be of immediate use. No disc copy is held once it is known that a secondary dump copy exists.

PERMANENT — This class includes files that are in regular use and for which it is appropriate to hold a copy permanently on disc. For security, a copy is also held in the secondary dump system.

File recovery

If the entire content of the disc store is lost, the operating system automatically initiates a major recovery operation and this operates in two stages: in stage 1 the system reloads all file directories and other administrative information, then in stage 2 the files are themselves reloaded. In order to reduce the time taken by stage 1, the incremental dump program writes one copy of the complete set of administrative information on to the beginning of each dump tape and the reload program will not recognise the existence of a new dump tape until this has been done. Individual file directories may also appear in other places on an incremental dump so the reload program must search for the latest copy. Unfortunately, however, the computer may fail during the dumping process itself and the information on the latest incremental tape may not always be terminated properly. For this reason, all data on a dump tape is self-identifying and the reload program can immediately recognise the end of the dumped information.

In contrast to stage 1, the second stage can be time-shared with the normal computing load, although it is usual to restrain other users until the communal libraries, at least, have been reloaded. Since the file directories contain the tape positions of the most recently dumped copies of all files, the system can initiate reload tasks automatically and a separate task is initiated for each magnetic tape that is needed. The normal workload scheduling algorithm decides how many of these tasks should time-share together.

It is fortunate that most system failures do not result in the loss of the entire disc store, but isolated losses commonly do occur. On these occasions the reload operation is shorter than, but substantially the same as, the more severe case. If the isolated loss involves a file directory then an abbreviated version of the stage 1 operation is required and this is time-shared with the normal computing load. The recovery of isolated files is handled by scheduling the necessary reload tasks but, for reasons of operating efficiency, requests for file recovery are usually batched and dealt with at regular intervals during the day (normally every 3 hours).

Requests for file recovery are not always originated by the system; the file owner may do this himself. There are three reasons why an individual may make such a request.

1. He has just accidentally destroyed a perfectly good file and he wants it reinstated.
2. He notices that a file has been corrupted (probably as the result of an undetected system failure).
3. The file is in archive storage and he now wishes to bring it into general use.

The use of the secondary dump as a means of obtaining archive storage has already been mentioned. Files classified as ARCHIVE are handled in the same way as others with the exception that the disc copy is discarded as soon as a copy is known to have been made by a weekly dump. The file owner can initiate recovery of an archive by re-classifying it as PERMANENT.

It is not uncommon for a file to be deleted in error and a file owner will occasionally wish to recover an old version of a file. The Cambridge filing system does not give the user a convenient means of distinguishing between two versions of one file; the user should do this himself by using similar but unlike titles for successive versions. However, in recognition of the fact that some people cannot organise themselves properly, the system prints a complete list of the files written onto the secondary dump tapes, and the careless user is free to inspect these lists.

The command

RECOVER file title dump identity

can then be used to re-instate the selected version of a file.

Restart points on disc

The two-level dumping system may be reasonably acceptable for user files and as a long-stop in the event of drastic failure. However, the loss of administrative information from the disc will usually result in heavy recovery costs and in this case the disc itself should be used to provide a third level of dump. One method of obtaining this effect is to write the administrative information onto a different part of the disc on each successive occasion so that the previous set of information is automatically available as a restart point. In practice, of

course, only two or three positions need be provided, but one must be sure to include some check that reduces the chance of overwriting all restart points with bad information.

By virtue of the fact that information on tape is self identifying and because the system does not seek to recover files dumped more recently than the administrative information which controls them, there is little danger of corruption due to confused administration when loading files from magnetic tape. To restart from information dumped on disc is a different matter since sequential updating cannot usually be guaranteed and information is continually being destroyed. Restart points on disc must therefore be planned with care and there are two techniques which can be used to reduce the chances of confusion during the recovery operation.

1. Artificial serialisation can be obtained by storing the time in each segment of administrative information whenever changes are made.
2. Interlocks can be imposed to guarantee serial updating of key information on the disc.

For example, a file may be deleted and the space may subsequently be re-used to hold other data. If the file directory containing information about the deleted file is not written on to the disc before the space is re-used, then there is a chance that, on restart, the new data will be attributed to the old file. The problem is solved by the use of an interlock that prevents the re-use of space until a new disc restart point has been established.

Error detection

Confusion in administrative procedure is the single most potent source of widescale file corruption; errors and inconsistencies here will propagate rapidly throughout the data base and can stimulate untold havoc. It is therefore this part of the base which must be checked most thoroughly.

There are essentially two aspects to error detection. The more conventional detection methods, of which parity checks and sum checks are examples, can be used to verify the integrity of a parcel of data while it is not being subjected to change. These methods are appropriate in any filing system and should be used to verify the integrity of individual files while not in use. However, the more awkward problem is that of checking the consistency of administrative information which is in continuous use.

Data will usually be handled in segments which are written to disc individually as occasion demands. A sum check, or similar device, can be used on each segment to verify its integrity, but additional checks are required in order to be sure that the several segments of administrative information together form a consistent set. Some checks should be made regularly under normal working conditions, but a very full check should be applied on restart after error. At Cambridge the start-up procedure, which is used at the start of the day and after an error, makes a thorough consistency check of the file directories and other administrative data. If inconsistencies are found, the administrative data is forcefully made consistent by such means as discarding files and file directories. Once consistency has been obtained and checked, the necessary recovery procedures are initiated.

The cost of detecting inconsistency can have an important influence upon system design and it must be considered at an early stage. For example, consider the possible methods of disc space allocation. It would not be unusual to find that the available space was divided up into fixed length blocks and that each file could occupy an integral number of these. It would then seem to be necessary to link together all those blocks that belong to one file and to form a single 'free store' chain of the remainder. However, if the blocks were chained through pointers held in the blocks themselves, it would be necessary to read every single block on disc in order to verify that the chains were all correctly formed; the process would be prohibitively expensive. A more practical solution is that adopted at Cambridge where a 'map' of the disc space contains as many cells as there are block positions on the disc. A file is represented by a chain of cells and the remaining cells form a chain of free space. Another method, equally satisfactory from this point of view, is to store a list of block addresses in each file directory and to use a separate map of one-bit items to identify free space.

Redundancy and system design

Wherever there is the possibility of inconsistency there must be some redundancy, and it is worth examining the role that redundancy plays in system design; few programs do not use redundancy but it is unusual to find a clear awareness of its existence. In the matter of system design it is valuable to consider each redundancy separately and to examine its role in the overall scheme.

Redundancy may perform three useful functions:

1. It may provide a check against error.
2. It may provide a means of recovery after an error has been detected.
3. It may be a means of obtaining some optimisation.

One would be well advised to identify both the cost and the effectiveness of each redundancy and seek a total system design that employs minimum redundancy to maximum effect. Redundancy provides an excellent source of programming errors and adds confusion to programming strategy. It will also be found that redundancy designed to serve one purpose does not always perform effectively in another role, and it will be convenient to handle each according to its function. For example, the start-up procedure for the Cambridge file system uses certain redundancies as a means of detecting error and inconsistency, whereas other information, intended for use as a means of optimisation, is reconstructed automatically and without question.

Errors in the recovery system

Errors which involve the dump and recovery operations themselves are particularly troublesome to deal with. The programs are, by their very nature, difficult to test and many of them are used only in times of crisis when one has neither the time nor the facility to deal with errors. It is therefore essential to write self-checking procedures into this part of the system and any time spent in this way will usually pay for itself handsomely.

The system will use magnetic tapes and other dismountable media so that successful operation can only come through close cooperation between the computer operator and the system. Unfortunately, in times of stress, humans are likely to take short cuts even when these involve substantial risk. It is therefore important to design a recovery procedure that offers resistance to unreliable operating procedures, but it should also encourage human verification of action automatically taken; a printed record of events is indispensable.

Many of the problems that must be faced when designing the dump and recovery system are posed by the possibility of hardware failure during the dump and recovery operations themselves. Each of these programs must be written in a way that allows the system to recoil from one error situation so that it can recover itself for another try. For example, the incremental dumper will need to be able to identify those files that are new or have changed and need dumping. One might therefore think that a simple flag is all that is needed to control this operation. However, if the latest dump tape is damaged the system will need to be able to backtrack and, in doing so, find all those files that will need to be dumped again. For this purpose, the Cambridge system records the tape name and block position in the file directory when a file is dumped.

The file reload program has its own similar problems. For example, a file directory may have to be reloaded from the most recent incremental dump tape. If, by some unfortunate chance, the tape cannot be read the system must ask to use a duplicate copy and then if both fail it must backtrack to an earlier tape. The backtrack itself means that the controls for the incremental file dumper may have to be re-set and this will have some effect on other jobs already scheduled to run. It is just this type of situation which is so difficult to foresee when planning a filing system, yet considerations such as these form an essential part of any bid to provide a smooth operating system and a high level of file integrity.

Acknowledgements

The Cambridge University multiple-access project was the work of a team working under the direction of Professor M. V. Wilkes, and was supported by the Science Research Council. It owes much of its initiative to informal contact with members of the Massachusetts Institute of Technology, Project MAC.

I also wish to acknowledge the assistance of C. A. Hoare during the preparation of this paper.

References

BARRON, D. W., FRASER, A. G., HARTLEY, D. F., LANDY, B., and NEEDHAM, R. M. (1967). File handling at Cambridge University, AFIPS Conf. Proc. 30 (SJCC 1967), p. 163.

DALEY, R. C., and NEUMANN, P. G. (1965). A general purpose file system for secondary storage, AFIPS Conf. Proc. 27 (FJCC 1965), p. 213.

Book Review

Conditional Markov Processes and their application to the Theory of Optimal Control, by R. L. Stratonovich, 1968; 350 pages. (*Elsevier Pub. Co. Ltd.*, £8 5s. 0d.)

The adoption of the state space description of systems has led to substantial advances in optimal control and filtering theory in recent years. Using this description the state $x(t, w)$ of the system and the (noisy) observation $y(t, w)$ of the state are Markov Processes. The basic problem in optimal filtering becomes the derivation of equations describing the evolution of the probability distribution of $x(t)$ conditional in the available observations $y(\tau)$, $\tau \epsilon [t_0, t]$, or the evolution of some risk function determined by the posterior distribution. The optimal control problem becomes the determination of a realisable control (function of t and the available observations) to minimise an expected risk. To deal rigorously with these continuous-time conditional Markov processes requires the development of mathematical tools of the same kind as those available for Markov processes, and this is one of the main intentions of the present volume. By contrast the theory for discrete-time systems is reasonably complete. Repeated application of Bayes' rule yields the conditional distribution of the state; the control problem is, admittedly, more difficult, but the dynamic programming technique can be applied, at least in those cases where it is possible to find a finite dimensional sufficient statistic (called sufficient coordinates by Stratonovich in a significant early contribution) for the posterior distribution.

The first problem to be faced (Chapter 1) is the relation between physical systems and stochastic differential equations. Related to this is the definition of stochastic integrals (Chapter 2) where Stratonovich defines an integral by means of central differences rather than by forward differences (as used in the Ito integral). Stratonovich's integral has the advantage of permitting formal manipulation by the ordinary rules of calculus, and leads, at least in the scalar case, to a simpler relation between the physical system and the stochastic differential equation. However, it has been shown by Clark that this apparent simplification disappears in general for the vector case. In addition the Ito integral possesses other advantages, such as the martingale property.

The central part of the book, consisting of the development of some basic concepts in Chapters 3 and 4, and the development of Stratonovich's main results for conditional Markov processes in Chapters 5–7, is written in an extremely abstract fashion. Most engineers, who are familiar with the underlying physical problems, will, like the reviewer, find the material difficult, if not impossible, to digest.

The remaining Chapters (8–11) consider various problems in filtering, detection and control theory. The examples are interesting and stimulating.

Stratonovich's earlier work was much more accessible to engineers. This volume, however, will be appreciated only by a few specialists. Nevertheless the problems considered are mathematically interesting and practically important, and this book should reward the perseverance of any reader with the necessary mathematical background. Stratonovich has been a major influence in the development of the subject.

D. Q. MAYNE (London)