THE UNIVERSITY *of* EDINBURGH

This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e.g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

- This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
- A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
- The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
- When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

BACKUP, RECOVERY AND ARCHIVING OF FILES IN A MULTI-ACCESS COMPUTING SYSTEM

Alexander Sinclair Wight

Presented for degree of Doctor of Philosophy

University of Edinburgh

1974

Abstract

General-purpose multi-access computing systems with files stored on random-access devices require that these files be protected. If the total on-line storage is inadequate there is a need for well-organized off-line storage. This thesis discusses the management problems involved in handling backup and archive copies of files.

In Part I we review what a number of systems, including the Edinburgh Multi-Access System (EMAS), have achieved. We also consider the influences of hardware and other forms of computing system.

In Part II we return to EMAS and propose a design and an implementation to provide comprehensive facilities, for backup copies of files and recovery of them, and also for archive storage.

TABLE OF CONTENTS

PART I

Chapter 1

Introduction

As computing systems have evolved and become responsible for ever increasing amounts of stored information, the problem of recovering from loss of some or all of this information has become more acute. In particular, multi-access systems with large-capacity on-line storage will provide an erratic and unsatisfactory service if this problem is not tackled. Since removable media provide cheaper storage than on-line devices, many systems will extend their storage capacity by using them. These can be used to provide additional copies of information and storage for information which is infrequently required. We call these backup and archive facilities.

The central material of this thesis is work on providing these facilities for the Edinburgh Multi-Access System (EMAS). A first detailed design was never implemented. What was provided was done after the rest of the system was running. So it was not part of a grand plan. We give detailed proposals for providing more comprehensive facilities. In addition we review how the same problems have been tackled both in similar systems and in different systems.

There may be situations where the problems of information protection and of handling a complex hierarchy of storage devices are important. This may be so to the extent of strongly influencing the choice of hardware and the system design. The level of automation

must be considered. A system could have fully automatic backup and archiving, such that it is largely transparent to the user. Alternatively one might provide tools for the user to manipulate his information himself. This could mean him being aware of the levels of the storage hierarchy or simply being allowed to arrange protection of his information with a degree of security that he desires and can afford.

In practice it is usually assumed that all related problems can be solved as programming exercises when required, or dismissed at the design stage with a few glib phrases about "usual magnetic tape facilities for backup".

If we list a number of the areas of computing system design where the designer might be considering facilities for backup, we have a number of areas of current and future active interest.

Data base management.

File systems.

Information integrity.

Information protection.

Information security.

Integration of batch and multi-access facilities.

Networks of computers.

Storage hierarchies.

This point of view sees a shifting of the focal point of interest in computing. Starting with programming, moving to operating

system design and now information systems, this interest sees the accessing and handling of large amounts of information as more important than computation. We use this argument to claim that the provision of adequate backup and archive facilities is an increasingly important topic.

Outline of Thesis

The rest of the thesis tackles the problem in the following way:

Chapter 2

We expand the material of this chapter to define the problem more specifically.

Chapter 3

This chapter describes how backup and archiving facilities have been provided on the Edinburgh Multi-Access System. It gives details both of how they were implemented and how users used them and the load this put on the system.

Chapter 4

Chapter 4 reviews reports of how other similar systems have fared in providing similar facilities.

Chapter 5

The user-support filing systems of the previous chapters have always been considered separately from data base systems. We examine the validity of this view.

Chapter 6

Operating systems other than general-purpose multi-access systems have a need for backup and archive facilities. A study of these may provide insight into the general problem of information protection.

Chapter 7

The provision of backup and archive facilities must be influenced by the media available for storage. Magnetic tapes have been used most often but discs and mass storage units may change the way information is protected.

Chapter 8

Continuing looking outward rather than considering backup in isolation we look at a number of questions which must be considered for any file system and see how they relate to accessing information on backup and archive media.

In Part II of the thesis we return to the design of backup and archive facilities for a multi-access system, EMAS in particular.

Chapter 9

This is the original EMAS design study. As described in Chapter 3 these proposals were not followed in practice, but are gradually being approached as the working system is improved.

Chapter 10

Drawing on the material of the previous chapters, we now propose new backup and archive facilities for EMAS. These are both the next step in the series of designs of Chapter 3 and the first step in providing a suitable environment for further experiments in backup and archiving for a general-purpose multi-access computing system.

Chapter 11

Following the experiences of Chapter 3 and system implementation experience in general, it is not advisable to implement all the proposals simultaneously. Chapter 11 suggests how the various stages might be done and introduced into service.

Chapter 12

Finally we review what has been achieved and what remains to be done in the provision of backup and archive facilities for a multi-access system.

Chapter 2

The Backup and Archive Problem

When information is stored there exists the possibility of it being corrupted, it may even be destroyed. Thus it is necessary to keep copies so that the lost information can be restored. We call the creation and recovery of these copies the backup problem. The problem arises in particular in a computing system where information is stored on-line on random-access devices. If the system has multi-access capabilities so that much of the stored information is changing rapidly then the adequate reconstruction of the on-line information in the event of a serious loss is a difficult problem. We consider this problem in detail. In addition if there is insufficient storage space for user's information on on-line media then provision must be made for additional off-line storage which users can access. The creation and use of this sort of information obviously have many similarities with that for backup facilities. Therefore we consider the provision of facilities for both services. Two extreme examples show some of the relationship between the two services.

1. If all copies of information produced for backup purposes are kept for the lifetime of the system then they contain the archive storage.

2. Where a system does not provide backup but gives users access to archive storage then they can provide their own backup copies by transferring information to the archive media.

Definitions and Relations to Other Aspects of File Systems

It has been said by Teichroew (46) that one reason for the lack of progress shown by research into computer-based file systems is that too many people have attempted to tackle the 'whole' problem. This thesis certainly does not do that. It describes one area where practical progress has been made and reports on continuing work on a particular project. However to consider all aspects of backup and archive it is necessary to look at their interfaces with other areas of information management. The fact that the following descriptions are informal and intuitive, rather than formal, definitions reinforces Teichroew's point. Perhaps an emphasis on practical work, as machine-independent as possible and widely reported, would lead to the development of both satisfying theoretical structures and frame-works for practical advances. This may need a diversion of effort from 'yet another scheduling algorithm' to considering what is being scheduled. From the backup and archive point of view it may also be desirable to work towards a knowledge of the items being handled. Information about the meaning and structure of a file could be used to provide a 'minimum' solution. There is no evidence so far that the theoretical work is either a suitable base for further advances or of value in guiding the production of more useful practical projects.

At the heart of all these generalities lies information, whether records, files or data bases belonging to students or corporations. Specifically we are interested in the storage of this information and

its security. Related to security we must consider integrity and privacy. The working definitions used in this report are as follows.

file - named body of information. Largest logical unit. Backup and archive is logically interested in naming and access and physically in storage and protection.

integrity - applied to the data in a file. Valid and uncorrupted. This implies hardware and software checking and error detection and correction. All measures to protect data are somewhat devalued if reliance cannot be placed on the integrity of it.

security - the protection of files. If we feel that the integrity of a file is guaranteed then it contains what we think it does. If it is also secure then even in the event of physical destruction we can replace it with a satisfactory known approximation or an exact copy of the original.

privacy - access control to files. Security is physical protection. Privacy controls restrict access. There is no consideration of the political issue of invasion of personal privacy.

data base - for many computing applications the relevant information is not a collection of unrelated files

of the sort described above but a very large number of identical records e.g. a collection of records containing information about a company's employees, and insurance company's customers etc. In such situations the whole data base must be protected - it may be thousands of megabytes. The unit which changes though is the record. In such situations a file now tends to provide logical access to either some of the records or some field from all of the records. We return to this topic in Chapter 5. The practical work reported in this thesis deals with protecting many independent files. The severe backup and archive problems are more likely to stem from data base implementations.

If the above can be called the concepts involved we now turn to introducing some of the practical details. As is explained in the next section many features of backup and archive storage stem from storage hierarchies i.e. a computer having its storage devices arranged in a hierarchy of increasing capacity and simultaneously greater access times and slower transfer capabilities.

One definition of backup and archive storage is simply the last level in this hierarchy. They hold files stored at this level for different reasons. Backup and archive facilities are not necessary

to run a computing system. Once they are then the initial provisions will probably be simple, crude and possibly effective. Also they come to be considered expensive in this form at some point as the system comes heavily loaded. It is at this point that the problem becomes interesting. A whole new range of questions has to be answered. Some or all of them may already have been considered when dealing with storage management for other levels of the hierarchy. But now the goal or measure of success may be different. Previously it was probably the effect on 'system throughput' or CPU utilisation, now the relevant units may be minutes and months rather than microseconds. Some of these interesting topics are:

data compression -

At the archive storage level it may very well pay to compress the stored data. This means finding a method of coding the data such that the amount of storage required is significantly less than in the original or standard form for the system. This means considering the cost of encoding and decoding the information. In turn this depends on the frequency of access to the data. Archive material lends itself to compression as the amount of storage is large and access infrequent. If the processing involved is largely low priority, that is using spare system capacity then the extra cost of decoding and encoding is negligible. Similar techniques have been used at higher levels in a storage hierarchy. (40).

Here there may be gains because the compressed data can be kept on a smaller, faster storage device and if the units moving between levels

of the hierarchy are also compressed then the traffic on the system may be reduced and again allow gains out-weighing the cost of encoding and decoding.

For backup as opposed to archive the situation is slightly different. If the backup storage is mainly composed of a copy of the on-line storage i.e. a checkpoint dump then there is little to be gained by compressing the information because unless on-line storage capacity is increased the amount will stay reasonably constant. However if the processing power of the system is greatly increased and backup copies of changes to on-line information are taken between checkpoint dumps then the total volume of backup storage may grow. When it is being dumped or read in a recovery situation there may very well be spare processing capacity so that encoding and decoding is not expensive and if for storage every saving is useful then compression may be worthwhile.

lifetime of archive material -

There are two interesting aspects of the lifetime of stored archive material.

1. Obviously as time passes there will be a growing body of archive material that is no longer required. Users cannot delete this material since it is off-line - they can only say they no longer want it. At some point the system administration must decide to physically delete all unwanted material and compact what is required. This means estimating the cost of saving storage space and processing involved.

2. Since by definition information is stored on archive media because it is going to be accessed infrequently some may be kept for many years. Little is known about the behaviour of magnetic tapes kept, for example, for greater than ten years and re-read. If there is compaction as in 1. then the writing of fresh tapes for material being kept will solve the deterioration problem.

Chapter 3

The Edinburgh Multi-Access System - Backup and Archiving

1. Introduction

Statement of Problem

The Edinburgh Multi-Access System (EMAS) and its associated disc-based file system have been described (50, 39). As with all such systems there are problems of loss of information from the disc and pressure on disc space as users' files expand. This paper describes how EMAS uses magnetic tape to attempt to solve these backup and archive problems.

The literature contains a number of excellent expositions of the problem. Wilkes (53) and Watson (48) describe the problem in general. Two detailed descriptions of particular cases are Fraser (21) and Considine and Weiss (17). Wilkes distinguishes user-support and data base systems. EMAS provides user-support facilities and although not explicitly providing for data bases handles files up to 4Megabytes. Contrary to the recommended approaches of the above, backup and archive facilities in EMAS were not designed into the file system. We describe how with this approach we have adjusted to changing system performance and user needs without having to tamper with the file system, which was also in a state of flux. We now propose to experiment with a new design which should bring major improvements and as always the user sees the system improving all the time.

Since mid 1971 the number of accredited users has grown from 50 to 500. As described by Rees (39) a user and all his files are assigned to one quadrant of the file system. Now (March 1974) there are three file system quadrants in use. Each quadrant has up to 80% of its 40,000 pages (each of 4096 bytes) allocated to user files. A quadrant caters for around 200 users. A user may have up to 120 files on disc depending on the size of his file index. This also dictates an upper limit on the number of disc pages his files occupy. Most users work within limits of 60 files and 1600 pages. There is no global disc allocation control except the archiving described below. Files may be protected or unprotected. The default mode is unprotected. If they are protected then the backup system keeps copies on magnetic tape. The magnetic tape facilities which have been used are 4 120k bytes/sec 9 track tape decks recording at 800 bpi on 2400' tapes. A full tape in the EMAS format holds around 4000 pages. A separate backup and archive service is organised for each file system quadrant. However the dumping programs can be run to deal with

a) the file system

b) a file system quadrant

c) a user's files.

It is obviously convenient for the recovery program to handle one more level i.e. a group of files.

The backup system has evolved through the following stages

a) dump all files daily

b) dump all protected files daily

c) dump protected and changed files daily, and all protected files weekly.

The archive system serves a number of needs

a) supplies cheap, secure storage

b) allows users to have more files than their disc index will allow

c) holds files which hav e been deleted from the disc because they have been unused for some time.

This helps to keep the allocated disc space balanced with the demand for more file space.

The archive system evolved through the following stages

a) dump unused, protected files

b) destroy unused, unprotected files

c) dump a file on demand (up to a week later in practice).

The next stages are to deal with the housekeeping of archived material as it expands and provide a service closer to backup and archive on demand, i.e. greater security but without overloading the system. Note that the total of backup material is much more stable unless another file system quadrant is brought into use.

There is a RESTORE command to allow users to retrieve files from archive tapes. This facility does not apply to backup tapes. The use of this command has been monitored to see the effect of a weekly archive dump and how often old archive material is used.

2. Users' View

The EMAS file system and standard user subsystem have been described by Rees (39) and Millard et al (33). This section describes the effect of the backup and archive systems on what the user sees. Files on EMAS may be protected by having copies made on magnetic tape. The default condition is unprotected. If a user wishes a file to be protected he issues the command

CHERISH (file)

HAZARD (file)

restores the unprotected state. This means no more dumps will be made. However in keeping with the current dumping philosophy no attempt is made to record the fact in a backup dump, so backup copies may still exist and the latest may reappear on the disc-file after loss of the current version. This may happen until all copies are destroyed as tapes are reused.

If information is lost from the disc file then the user may have lost unprotected files. For a protected file the restored copy may be up to 24 hours out of date. There is no automatic way a user can request a file from the backup tapes.

When the user finds files missing from his file index then the archive system has been at work. If an unprotected file has been unused for four periods then it is destroyed. In practice a period is a week. Similarly for a protected file, but a copy will have been made on two magnetic tapes. This also applies to files for which a user has requested archiving with the command

ARCHIVE (file).

To combat or cope with this situation the user is given two more commands.

FINDFILE (file)

allows him to enquire about his archive material (whether requested or automatic) and

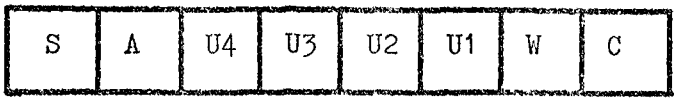
RESTORE (file)

puts a copy of a file back on the disc and adds the name to the user's file index. The output from FINDFILE can be directed to a file so that the user can manipulate it and display it in forms other than the chronological ordering supplied. A user can also ask the administration to write a private tape.

In the case of files which have been permitted (39) to other users then these 'permissions' are dumped to tape with the file. A file restored from a backup tape has the 'permissions' restored. This is not done for an archive file nor does the fact that a file is permitted prevent destruction or archiving.

3. Backup

Each file system quadrant on EMAS is backed up independently



Meaning for each flag if set.

C	protected
W	file connected in 'write' mode
U1	file connected i.e. used in current period
U2,U3,U4	usage over previous three periods
A	request for archiving via ARCHIVE command
S	spare

Figure 1 - ARCH byte of file index entry.

The backup program is run in an executive process (50) under operator control. This dumping is done with users running, but overnight when the load is light. Any files in use which are open for writing are ignored. On a daily basis those files created or altered since the previous day's dump are copied to magnetic tape and the 'written-to' flag in the ARCH byte (Figure 1) in the file index entry reset to zero. In addition once a week a dump is made of all protected files. If this quadrant is lost then all protected files can be put back by reading the daily dump tapes back to and including the most recent weekly dump in reverse chronological order. Copies of files other than the most recent are ignored. As a further precaution a number of these weekly cycles are kept. Fresh tapes are written each day. Tapes are not mounted with a write ring while they contain valuable information. After a dump the tapes are read as a further check. If daily tapes cannot be read then a complete weekly dump of all protected files is made. A list of what is on a tape is produced and the files on tape can be completely identified by reading the tape. No other records are kept nor are the file indices dumped. They record only what is on the disc and have no usefulness on tape until we decide to record all the changes. When restoring from tape rebuilding the index is very simple. No record is kept of a user destroying files so recovery may see 'dead' items re-appearing on the disc. In the same way (see Users' View) permissions which have been revoked may be set up again if the permitted file is recovered from a backup tape. In the next more

flexible versions of this system these situations will be improved.

4. Archive

The archive system is run in exactly the same way as the backup system but using more bits of the ARCH byte to drive it. Ideally it is done immediately before a weekly backup dump to prevent ARCHIVE material reappearing in a recovery situation. If the archive bit for a file is set then the file is copied to tape. The usage information on which the other archive actions are based is generated as follows. The rightmost bit of four is set when a file is accessed. Once a week, or whatever 'period' is chosen, these four bits are shifted left. So if a file is not used for four complete periods these four bits will be '0'. The archive system destroys unprotected files with this pattern. Cherished files with the same pattern are copied to tape. Once the file has been copied to two tapes, a record added to the index of archive material and a line-printer index of the newly dumped material produced then the disc copy of the file is destroyed. Each archive run starts with fresh tapes. Material from previous weeks is not put at risk by mounting the tapes with write rings again. This obviously results in wasted tape space, especially now as we move to 1600 bpi tapes. However the flexibility demanded for other reasons (see Section 7) and attempting to satisfy requests for archive on demand means that this problem must be solved.

The archive index is a file owned by the MANAGR process. The FINDFILE and RESTORE commands access this index on behalf

of a user to list any entries required and to find their tape addresses. RESTORE sends a request to the VOLUMES process to have the appropriate tape mounted, the file read if the one found at the tape address has the same identification as the requested item and the name added to the requesting user's file index.

It has turned out without any 'tuning' that this system leaves each file system quadrant in a balanced state i.e. each week the space created by archiving is sufficient to hold the files RESTORED and created.

5. Implementation

The programs for backup and archive run as part of the privileged executive processes MANAGR and ENGINR. So two file system quadrants can be dealt with simultaneously if enough tape decks are available. For the period covered by this report we have had 4 9-track, 800 bpi, 120K bytes/sec decks. See Section 7 for the effects of new hardware.

The following data is used.

1. List of users.
2. Each user's file index.
3. File belonging to MANGR which contains an index to the archived material for this quadrant.
4. Date and time supplied by system.
5. Tape identifiers typed in by operators.

Apart from the tapes written, output is the updated archive indices, teletype monitoring of the running program and a line printer record of the files destroyed and written to tape.

A file is written to a tape as a CHAPTER. This is the standard EMAS tape format. A CHAPTER is an 80-byte header and a number of pages. A page block is actually 4120 bytes (4096 data + 24 identifier). The tape is addressed as chapter and page within chapter. The backup and archive programs put in an extra page of information as the first of the file. This contains as much identifying information as possible and the list of permissions if there are any. The average file written to tape is 8 pages. A 2400' tape holds up to 4000 pages. The maximum size of a file is 1024 pages. We do not split a file across tapes. Separate tape sequences are maintained for each file system quadrant and we do not add to tapes at the next dump so the average tape is around half full. See Section 7 for changes under way.

The, thankfully very rare, job of replacing a complete file system quadrant is done by reading the backup tapes in reverse chronological order up to and including the most recent weekly dump. If an individual file is required from a dump tape the material will be read from the tape position derived from the line printer records. All the standard programs will only hand over a tape file to its owner as recorded in the extra page with the file on tape.

For archive recovery the FINDFILE and RESTORE routines are part of the user subsystem (33) and use the information stored in

the MANAGR files. These files are connected in shared mode in the user's memory. If a user were to detect this he could gain access to other user's archive records. Strictly this is a breach of privacy, and the information should be handled behind the system interface as represented by DIRECTOR (39). There are separate mechanisms to dump the supervisor and MANAGR files.

6. Operational Experience

The backup and archive system described above has been in operation for 16 months. It was preceded by a much simpler one and will be followed by a more comprehensive and integrated one. This section describes how the system has coped with the demands made on it.

Users protected one half of their files. With an active population around 500 holding 70,000 pages spread over 3 file system quadrants this generated a weekly checkpoint dump of 35,000 pages, (140M byte). The daily dump of new and changed material on the five working days (a weekend service was not a regular feature) was around 5000 pages (20M byte). In practice this material has seldom been used. Tapes are re-cycled after a few weeks. No copies are removed from the building.

The archive system has been generating tapes for 16 months. None of this has been discarded, although users can mark files as no longer required. On-line directories are kept. The material extends to 900M byte and the directories occupy 1.4M byte (roughly

equivalent to the on-line file store of 3 users).

An average weekly run of the archive program destroys 2,500 pages of unused, unprotected material and transfers to tape, 5,000 - 7,000 pages. About 80% of this is unused and protected. The remaining 20% has been requested by ARCHIVE commands during the past week. This figure is small and tends to be dominated by one user in any week. The result is to free up to 10,000 pages of disc space to cope with new and extended files over the next week.

The RESTORE command was monitored for a 3 month period to find what use was made of the archive material.

Number of users issuing requests	287
Average number of requests/user	14
Average number of requests/day	60
Average size of file restored	12 pages
Average time between archiving and restoring (i.e. 90-100 days since last used)	64 days
70% of requests referred to files dumped in the previous month	
40 requests were for files over 1 year old	
20 separate tapes were required each day	
Average time from RESTORE command to file available	5 minutes

This is the aspect that users like most. It is convenient to have the system doing file housekeeping for one and yet be able to retrieve migrated items very quickly. Obviously some users write programs to access all their files and ensure they remain 'in use'.

As a sidelight on the control that archiving applies, when some users were recently transferred to the second 4-75, and initially archiving was not done, one file system quadrant (150M byte) was full within a month.

7. Historical Development and Planned Improvements

EMAS as planned by the EMAP team (50) was to have an elaborate backup and archive system under the control of processes activated whenever action was required. This would have provided full checkpoint and incremental facilities. However the first working file system was simple and the backup and archive was restarted to develop in parallel with this and a user service.

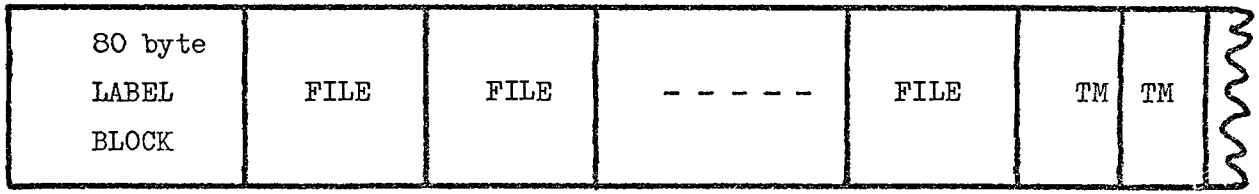
The following were the development stages.

1. Copy the half disc-file in use to the free half.
2. Dump daily for each user his listed protected material to his own tapes.
3. Dump daily for each file system every protected file.
4. Change the frequency of 3 to weekly and add a daily dump of created and changed files. This is the present system.

It is now planned to implement backup and archive in the style of the original proposals. The situation will be made more complex by the extended configuration of two 700M byte disc files, two 4-75 processors and a Front-End Processor. The present system provides two levels of checkpoint dumps. This means a user may wait up to 24 hours for a dump and a week for an archive. This is too long. An improvement would be to dump more frequently and archive requested and unused material on different cycles e.g. daily and weekly. We have always aimed to keep the load generated by these services low. Certainly the reliability of the disc-file has been a great help.

Apart from providing an incremental service and implementing it with a privileged process activated by user request, or the system on an alarm clock basis, the major changes are to cut the number of sets of tapes, so that those in use are filled, and to maintain a dump and archive index for each user. This will be maintained by the backup process and exist as another level of each user's current File Index and be moved with him if he is transferred between file system quadrants for administrative reasons.

This means that a data base as opposed to user-support problem must also be solved, i.e. the many changes to the records in the new index must be very secure. The backup for these changes is therefore very important. Tape material will still be self-identifying so that on-line indices can be reconstructed.



TM = Tapemark

FILE = TM, 1 page identifier, TM, file in page blocks

Figure 2. Tape Format

One final problem is the control and compression of the volume of ARCHIVE material.

In the new situation there are going to be new tape decks and new tape handling software. Instead of 4 decks and 1 processor there will be 4 1600 bpi 120k bytes/sec decks accessible to 2 processors and supervisor will provide tape-handling primitives rather than a specified format. The dump and archive system will use the simple format shown in Figure 2 . The header will contain all the information previously held in the various identifying blocks. This change provides a convenient time to 'lose' all unwanted archive material. This may point to the cost-effective solution to the problem of archive explosion in general. Organizational and administrative considerations will outweigh any algorithmic results based on charging for space whether by explicit allocation or using expiry dates. Note that although the archive index may continue to grow this is not very expensive as there is no automatic search of it if a specified file is not found in the user file index.

Conclusion

The previous sections have described a backup and archive system for a user support environment. It has grown to match an evolving system and user population. Having seen what users need it can now be changed to give an improved service in a more complex system situation. This would appear to be the best way to do things.

The above does not solve the problem in other systems, e. g. small configurations, RJE systems and any genuine data base systems. Again an evolving system to match usage will almost certainly be better than one aimed at coping with all possible situations.

Chapter 4

Other Similar Systems - Backup and Archiving

An appraisal of reported work on backup and archiving in EMAS-like systems.

Introduction

The previous chapter described the use of magnetic tape to solve the backup and archive problems (as defined in Chapter 1) in the EMAS system. As mentioned there other workers have reported on various aspects of these problems for other similar general-purpose multi-access systems. Before proceeding with the design of a more comprehensive and elaborate solution for the EMAS case we consider in this chapter other work of a not too dissimilar nature.

Before making detailed comparisons we must discuss how to make them, how much weight to attach to them and how to evaluate their usefulness in understanding more about the problems and in supplying information for future implementations. One way to tackle this is to work backwards from the ultimate aims. As stated in Chapter 1 these are not to supply a design which will solve all recovery and archive problems and be suitable for any system and environment. However it seems reasonable to aim for a design which is

1. Modular.

   This implies two things

a) separate components to handle specific problems i.e. any particular implementation need not contain every component

b) primitive functions at a lower level. These will be the basic building bricks or services which the components in a) will use.

2. as self-contained as possible. This means that the interfaces of the backup and archive package with components of its environment are well-specified and useful in transferring the same package to another environment or implementing a similar package. Another way of stating this is that the primitives of 1.b should be obviously implementable in the largest possible range of systems.

If these aims were to be achieved they would go some way to satisfying proponents of both implementation schools. These state that

a) backup and archive must be designed as an integral part of the file system

b) backup and archive are so dependent on system performance (hardware and software), and user behaviour, that they should be provided and developed in a tag-on manner which does not interfere with any other components of the system.

Pursuing the use of package above a bit further these design aims can be interpreted not as an attempt to provide a package which will do everything for everyone but as an attempt to tell people how to put together their own tailor-made package and be able to discuss it with people in the same situation.

Without at the moment trying to identify these primitives, modules and interfaces we list the following topics as suitable headings under which to compare some backup and archive systems.

1. backup, archive, copy, restore and recovery.

   Backup and archive are defined in Chapter 1. Copy includes making on-line copies as well as copies on removable media. Restore is the inverse of copy i.e. replacing an original, possibly corrupt or null, with a previous copy. Recovery is the identification of an error state and the actions required to create a satisfactory state. This may involve a restore of a copy as a very simple example.

2. file system, user population and user behaviour.

   We do not attempt a formal definition of file system. A file is the largest unit of information common to both users and the functions of 1. In comparing backup and archive systems we must consider also the use made of the file system by users, i.e. file activity. This means looking at

number of users

number of files

access patterns.

An upper bound on the volume of traffic generated may be estimated from these figures.

3. file storage.

We are interested in the hardware devices involved and the administration of them. Specifically whether a hierarchy is involved and if backup and archive media are considered part of it. The administration of the devices may be automatic or under management control or some combination of these. The backup and archive system may be part of the controlling mechanism or invoked by other managing authorities. In particular how is the total volume of archive material controlled?

4. user and system commands.

Commands to a backup and archive system may be explicit or implicit.

The Cambridge System

Fraser (21) has described the backup and archive facilities provided in a disc-based multi-access system at Cambridge University. The description is very complete, to quote the discussion in Hoare and Perrot (21) "a very thorough paper, so precise in detail that it is almost a guide to implementation". This justifies looking at it

closely with the stated aims of this chapter in mind. In particular this work was done before 1970, yet we have the Multics proposals of 1966 (15) and details of what 'happened' or went wrong in the following 8 years (45). As Chapter 3 showed EMAS has not been immune from the same problems that Multics has encountered.

1.

backup

Newly created or changed files are dumped to magnetic tape. There are two parts to this system. The incremental backup system runs frequently, perhaps every 30 minutes or few hours, and dumps all files created or changed since the last run. In addition once a week a dump is made on a separate set of tapes of all protected files. These dumps look like the EMAS weekly and daily dumps but the organisation is different. There is one pool of tapes used cyclically for the incremental dumps. For the secondary dumps users are organised into groups such that their dump material will fit onto one tape. A grandfather, father, son cycle is then used for each group. In addition the users are also grouped so that each group gets a weekly dump of all protected files, but they are not done at once. Twelve separate dumps are scheduled throughout the week.

archive

The archive system is the secondary backup dump. Archive files are kept on these tapes either because a user has requested

that these files should exist off-line or because he has filled his disc allocation. There is still a directory entry for the file on the disc. So these tapes are both an extra level in the storage hierarchy and long-term storage. The dumping is either for security or redistribution of space in the storage hierarchy. The primary and secondary dumps are integrated in the sense that they are really two incremental dumps one fast and one slow. We can consider the secondary dump incremental because it is largely tape to tape with only incremental material from disc. However the two systems do not communicate or use each other's information.

copy

There is no copying done on the disc. For both dump systems the same copy software is used to transfer a file from disc to tape. The archive system requires extra code to cope with tape to tape copies.

restore

When a file is missing i.e. required to be copied from tape a flag is set in core indicating that the appropriate file directory needs a restoration done. Once an hour the flagged directories are searched and a list of missing files compiled. The required dump or archive tapes can then be loaded and the files restored. There is no automatic mechanism to search for a second copy if a reload fails. As Fraser says, such a facility would be useful but difficult to implement because of the problem

of automatically classifying errors and taking sensible action. There is another important point to be noted here. Directory entries exist for missing files. These are preserved if a restore fails. This is to be distinguished from failing to create a file which would result in no entry. A failed restore means simply that the file is still not on-line.

recovery

The Cambridge system includes a lot of checking and redundant information. After a failure the file data base is checked for consistency and any inconsistent information ignored. Jobs to restore any missing directories and files are simply scheduled into the normal queues. All directories are put on the front of every dump tape to cut out search time. Note that there is very little tape searching or scanning in this system. Once the directories are up to date from the current dump tape every recorded file has a unique tape address whether as an archive file or a dumped copy of an on-line file.

2.

file system

The file system contains 10,000 files. There is a two-level directory structure with one entry for each user in a master directory and an entry for each of a user's files in his directory. Again this is similar to the EMAS system and is to be contrasted with the deep hierarchies of Multics which appear to have caused trouble in the backup system (next section). As Fraser points

out the two level system is sufficient for addressing and not necessary for any other reason such as security or accounting. These can be achieved by other means.

user population and behaviour

There are 700 users owning 10,000 on-line files.
6000 characters suffices for the average user file. This has strongly influenced system design so we need not look here for a solution to the problems of commercial data processing.
A file consists of an integral number of 4096-character blocks. The maximum size of a file is 100 blocks. These figures indicate that even with active users it is sufficient to use a file as the dump unit and the volume of traffic from disc to tape will never be outrageous.

3.

file storage

The on-line storage is on a disc with a capacity of 128 million characters. The file system uses 80% of this. The view of magnetic tape is not that of an automatically controlled level of a storage hierarchy but again like EMAS the user is aware of its existence and is given means of exploiting it. The Cambridge system of archive storage appears to tend more to user control with the system able to override it. EMAS gives the system more control but keeps the user informed and allows him override as well.

4.

commands

There are no descriptions of any of the commands available. Just two statements. Command systems tend to be somewhat parochial. Users make use of the archive store simply by classifying and re-classifying files i.e. changing permanent to archive will cause a dump and archive to permanent a restore. There are limits on the number of files in a directory and the amount of space that Temporary, Permanent and Archive files may occupy. These again make the user more aware of the file storage system.

The Multics System

As reported in Whitfield and Wight, (50), there have been many similarities in the development of Multics and EMAS. In this section we explore how this has also applied to backup and archive systems. The brief outline for both Multics and EMAS consists of

a) early detailed plans

b) lowering of priority under pressure to get something working

c) an interim scheme

d) a series of ad hoc improvements to the interim scheme

e) a stable system and pressure to 'do it properly' with an implementation of something akin to the original plans.

The major difference appears to be that the interim EMAS scheme has been more successful in meeting system constraints and fulfilling

user needs. We will discuss this point further in the proposals for the next EMAS backup and archive system. The analysis of the Multics backup and archive system in the following sections is based on the design proposals of Daley and Neumann (19) and Clancy (15). Progress after 7 years was reported by Corbato et al. (18).

The up-to-date information comes from Stern (45)

1.

backup

The design aim of the Multics backup system was to provide 'high storage reliability' by copying 'new data placed on secondary storage onto some detachable and preservable medium'. This was to be done by running periodically an incremental dump program. This would copy to tape all new files and all files changed in the period since the last run of the program. Also changes, deletions etc. would be dumped. If this material were kept for all time then all forms of restart and recovery could be provided and information loss would be restricted to the period of the dumper. Two further dumps were proposed so that there would not be intolerable delays when reloading after a crash. These were a system checkpoint dump and a user checkpoint dump.

a) system checkpoint

This would dump a suitable base of system and accounting material. This would be defined as that material required to open a service to users. (See recovery).

b) user checkpoint

To speed the reload of material belonging to the most recently active users a dump would be made in the same way as the incremental one but with a much longer period. (cf. weekly dump in current EMAS. It is not quite the same). (See recovery).

archive

There was no Multics proposal for a separate archive system. An automatic system would arrange to have material transferred off-line if there were pressure on space on on-line devices. However almost certainly no transfer would be required because the actions of the dump programs above would ensure that an up-to-date copy already existed on tape. So in this system backup and archive dumps are integrated and the only difference is that an off-line archive file still has an on-line reference but backup files are unknown and inaccessible to the user.

copy

The copying of files for backup purposes would be done by a background daemon or process. This would always exist and be automatically scheduled to act by scanning the file system at the appropriate periodic times. Archive copies as pointed out above would not involve copying as the files wo uld exist on tape already, via the incremental dumper.

restore

All of the writings on Multics give the impression of concentrating on solving the problem of totally restoring on-line storage in the event of a catastrophe (recovery). Thus there is little discussion of the possibility of restoring individual files after minor mishaps. Stern (45) does discuss it but as a proposal not a facility.

This would appear to indicate that for protecting total storage, prevention is more important than cure. If this is coupled with an efficient restore service then system availability and user satisfaction are higher. Certainly for a general purpose multi-access system a degraded service is not satisfactory. If the system is down completely then the length of time that it is down is not so important. But sorting out minor losses should not inconvenience a lot of people. The implications of Stern's remarks are that Multics users suffer twice over. Because there is no provision for restoring material after minor losses a complete recovery must be done. This takes a long time. Therefore to save reload time a special dump is done every second day. In this dump a simple page by page copy of on-line storage to off-line media is done. The logical structure of the file hierarchy is ignored. Obviously for this dump the service to users is closed. Two conclusions are that, for any backup or archive design, whether it be an overall plan or an interim scheme, if attention be given to

identifying errors, and the subsequent recovery or restores required, and to minimising the need for a complete recovery then the time spent doing so will be amply rewarded. This obviously has implications for the file system design and file storage devices chosen. Extreme cases are systems which cannot tolerate disruption of service and so cannot afford complete recovery and those with small on-line storage where everything can be put back quickly so a total dump makes sense. However as Stern (45) points out the expansion of on-line storage is greatest problem likely to trouble a working backup system.

2.

file system

There are two important points about the Multics file system as far as this discussion is concerned

1) It is organised as a tree-structured hierarchy of directories and files. Files appear as terminal nodes of this tree. The name of an entry in the hierarchy is unique within its containing directory but to uniquely identify an entry in the total hierarchy we require access to its pathname i.e. the ordered sequence of other parts of the hierarchy of entry names which describes a path from the root to the desired entry.

2) The file system records the information covering the lifetime of the system. So an extant file retains a

position in the hierarchy. If it only exists in off-line storage, archived in EMAS terms, this is simply a difference in the directory entry recording its storage status. There is no other special provision for archive status. No on-line record is kept of where backup copies are kept. Stern proposes a system allowing user-retrieval of backup copies in a manner similar to the EMAS archive system. A more general approach to the whole question of access to off-line information is detailed in Chapter 5.

This file system structure obviously complicates the backup situation considerably. In recovery situations it is not simply a matter of restoring for each user the files belonging to him. A complete hierarchy must be built up and provision must be made for dealing with failures when restoring directories. Directories cannot just be restored they must be kept consistent with both the inferior and superior entries in the hierarchy. Stern has described in detail how a Multics implementation might handle these. This hierarchy structure also has implications for the use of file storage.

3.

file storage

The main point about file storage to emerge from Stern's thesis is that the intuitively appealing approach of having one file system hierarchy mapped onto the secondary storage devices

and the transfer between devices being determined by program to keep the most accessed material most readily available is wrong. If a directory is on one device, and the files or directories corresponding to its entries on another, then if the directory device is lost all trace of the files has gone. Also there is no record of what was on the lost device. This reinforces the arguments in Chapter 3 for the EMAS system having separate groups of users. Arguments that storing separate parts of the hierarchy on separate devices is inefficient, because each device must have unused storage space to allow for expansion of that part of the hierarchy, are not convincing. Dealing with dynamic objects like directories and files it makes sense to have spare space on devices rather than generate extra traffic between storage devices. So this is one point where the backup system should influence the file system. The idea of one hierarchy must be questioned. We want the file system to be structured and stored so that the case of a complete recovery of on-line storage is very rare. This need be put no more strongly than that in general the file system should be modular and stored on separate identifiable storage modules. These do not have to be removable or independent.

4.

commands

The Multics user sees very little of the backup and archive

system. However Stern has an interesting proposal for what he calls user-controlled backup. This extends the discussion in two ways.

1) Consideration must be given to whether periodic dumping matches the rate of change or importance of changes to a file

2) Can we ensure that a dumped file is 'consistent' i.e. it is meaningful and useful if it has to be restored.

We treat 2) first. According to Stern, Multics ignores the problem of consistency and dumps files regardless of whether they may be in the process of being altered. He argues that for most common changes, i.e. editing and the production of compiled object code, this does not matter as these tend to work by accumulating changes in a temporary area and producing the final modifications to the original in one continuous sequence, rather than as specified by the user. In this way it turns out that few inconsistent copies are dumped. Obviously EMAS avoids the problem completely by currently only dumping once per day and then a file is not dumped if it is connected in write mode. However if a system dumps more frequently, and a file is accessed often, or open for long periods, then it may not be clear what a restored copy represents. Even the date and time of dump is not enough if the user has no exact record of when his changes to the file were made. The extreme examples of this problem come from the sort of data

handling common in commercial applications where a file is permanently open and data records are being changed. An arbitrary dump of this file cannot reflect any state which is more important than a random one. We consider this problem more fully in Chapter 5. However we can note here that in an installation devoted to this form of computing all changes would probably be logged so that any state of the file could be reconstructed. For a large file, dumping the complete file every time it changed would appear a rather crude solution - if it were possible. Both Stern and Chapter 10 propose solutions to the situation where a general-purpose computing system has to cope with backup of a 'transaction' file where simply noting change and dumping copies of a file are not good enough. The user must be given some control or the system must give itself the ability to make use of his special knowledge. It is obviously possible to have the same arguments as occur about the usefulness of user-supplied information in main memory and processor scheduling. However in our case the user would appear to have more useful knowledge.

3. TSS/360 at T. J. Watson Research Centre archive

Considine and Weis (17) have described a system of handling 'migrated' storage that is similar to the use of archive storage on EMAS. Further information on the use of coding techniques to compact this storage is reported by Katcher (29). This is

discussed in Chapter 8. The scope of the system discussed appears equivalent to one EMAS file system quadrant i.e. around 30,000 pages of on-line storage and up to 200 accredited users. The migration of material to archive storage was carried out at monthly intervals. One major difference from anything we have considered so far is the use of two levels of archival storage. Archive material may be stored on direct-access volumes or tape. The whole system is based on a standard TSS/360 facility for tidying on-line storage. The two variations are

1) to create, as well as cleaned up on-line storage, off-line copies only of material not used since some date
2) to copy only unused material. This version may be used to process off-line direct-access volumes to produce tape archives.

An on-line file is kept recording the disposition of these archive files. An entry is 104 bytes.

restore

A user can list his archive material and ask for restoration of a particular file to on-line storage. In this case the archive copy is erased, literally for a direct-access volume, pending for tape. This seems a somewhat wanton squandering of redundancy!

file system, user population, user behaviour

The interesting fact from this section is the amount of archived material put back on-line. The figures are

pages on-line	32,000
pages archived	19,400
pages restored	900
pages erased	1,100 (by explicit command presumably)

The authors claim these as a success and a vindication of their policy. However it appears to us they would have been more successful with a simple file classification scheme. It appears more likely that their off-line volumes contain unwanted temporary files rather than genuine archive material. This is a view confirmed by the fact that the total storage ownership profile is very similar to that for on-line storage. One would not expect long-term archive storage to exhibit this pattern.

commands

In addition to those already mentioned a user can specify that a file should be included in the next archive dump and also that an archive file be deleted. There is a system command to attempt to tidy up archive storage, deleting files from direct-access volumes and checking for tapes with no useful material. The authors observe (as in EMAS) that a small number of users use a large proportion of the on-line storage and that this should be taken into account, as well as the time since the file

was last used. While agreeing with this point we do not feel that their stated intentions of making their scheme more 'sensitive' to many more factors and transparent to the users is necessarily ideal. By all means hide it from the small user who does not upset the system but the large user should have some controls, he may not intend to be a large user. Also there is a danger in responding to every variation in storage allocation. The aim should simply be to stave off collapse. This does not include reacting to all temporary fluctuations.

There are three more papers we consider worth mentioning here.

The Chilton Multi-Access System

There is a report on this system by Thomas and Baldwin (47). As a tail-piece to the section describing the file system they point out that space was left in the directory to implement an incremental file dump system. But it was never used because the hardware was so reliable. A full weekly dump proved sufficient. They allow the recovery of individual, even deleted, files from these dumps. An improvement would have been to implement the incremental system even if not to use it. It might have been useful some day, adding or changing storage devices for instance.

The Hatfield PDP-10 System

Mitchell and Holmes (34) describe a disc-based backup system

- disc-based because they had no money for tape devices. The resulting system appears to have been very unsatisfactory. Mainly it would appear because of problems with the file system and an inflexible approach.

Data base software: a sceptical viewpoint

This paper by Gilb (23) has one point worth noting here. Gilb offers the suggestion of reverse dumping for certain situations. This means keeping the data base off-line on magnetic tape and periodically (perhaps daily) load it onto the on-line devices. The tape version can be simple and standard. The on-line working version can be quite incompatible and organised for optimum use in the particular environment. If updates are logged there is complete backup. Obviously this is not generally applicable to all general-purpose multi-access systems but indicates a freshness of approach, though the idea is not new, which might be stimulating for particular parts of a file system or in particular environments.

Chapter 5

Backup and Archiving for Data Bases

In previous chapters we have discussed how backup is dealt with in a number of general-purpose multi-access systems. In all of these systems it was assumed that the most convenient unit for dumping and recovery was the file. Any changes, however small, could result in a file being dumped to tape. If information was lost then it could be restored, either one file, a few files or all on-line storage. It would be done on a file by file basis. Wilkes (53) calls this a user-support system. Two obvious points are that 1) no system attempted to guarantee completely up-to-date recovery; any changes in the previous hour, few hours or day could be lost, and 2) no files were so big that dumping them alone might take hours.

These demands might be made in other computing applications particularly in the business world. Wilkes calls these data base systems. Backup for data base systems has been mentioned in passing in data base literature over the years. However as Canning (6) says, "We have not read or heard much about backup provisions for large data bases".

As pointed out earlier, we think progress will be made tackling specific file problems. The whole subject of data bases is beset by problems and pitfalls. Therefore we propose first to consider the dumping, restoring and archiving of very large files. Any solutions must then be evaluated in practice to cope with particular data base

implementations i.e. to cope with:

1) logical structure of data
2) storage structure of data
3) the rate at which transactions access the data base
4) minor recoveries
5) complete recovery
6) being used for reorganisation purposes i.e. to improve storage and access
7) existing inefficiencies and not generating too many new ones
8) automatic dump and recovery
9) user controls
10) managerial controls.

History

For user-support systems random-access storage has replaced trays of cards or similar media and backup is provided on magnetic tape. Data bases however started on magnetic tape and in this situation backup is simple and straightforward. That part of the data base represented by a particular tape can be updated by mounting the tape in read-only mode, so that data is relatively safe, and copying it onto a new tape, incorporating any updates currently held on some other medium. By keeping the tapes from a number of these cycles (traditional grandfather, father, son system) and also the updates then a very secure system is available. However, once the data base moves to random-access devices two problems arise:

1) instantaneous updates can be easily done so a daily dump is not good enough

2) records in the data base are no longer dealt with sequentially. There may be structure, reflecting relationships between records, and extra information to permit rapid access. Protection of this structure is also required. The problems of Multics with a hierarchial file system, described in the last chapter, are of this form.

Periodic dumps and journals

The traditional method of dealing with this problem has been to use some combination of periodic dumps and the recording of journal tapes.

Periodic dumps

A periodic dump is a complete copy of the on-line data base made on magnetic tape. If it can be done in minutes or a few hours it may well be done daily. If the data base is very large this dump may take an intolerable time e.g. 12 hours or more than a day. Although if this was to be done, it would be sensible to do as much dumping in parallel as possible. This dump will be required most dramatically if a complete loss of on-line storage occurs. It may also have to be searched or accessed for individual items lost. Another use of this dump would be to reorganise the data base. If the updates to a data base include addition and deletion of records as

well as just changing records in situ, then the average access time may rise because of more complicated searches. The problem is then to compare the cost of less efficient access against the cost of a reorganisation involving a re-reading of a periodic dump. This is a managerial problem and should be invisible to users. Shneiderman (44) presents an analysis of this problem. The maintenance of a data base on magnetic tape provides this facility. The old version becomes the backup while the "dump" is the new well-structured version. Obviously with enough on-line storage but split between two or more sites, the same technique could be used. A "dump" or second copy made on-line would provide both backup copy, which is the current one, and a reorganised copy which is now the current one.

Journal tapes

Between periodic dumps, or without them, backup can be provided by recording on magnetic tape the changes to the data base as they occur. These may be recorded in a number of ways depending on how they are to be used.

a) Before

A before journal contains copies of records before they are changed. These can be used to step back if changes turn out to be faulty.

b) After

An after journal contains copies of records dumped after they have been changed. These can be used if on-line copies are lost. It is obviously desirable that some on-line record of

these copies be kept, both to minimise searching for individual records and to allow compaction of these journal tapes. This would happen if periodic dumps were few or not taken. In this case the most compacted form of the continuing journal tapes would represent the backup of the data base.

c) Transaction

A transaction journal recores the "commands" to change records. This means that a series of changes can be repeated starting from the same base and using the journal as input rather than the command streams generated by operators or programs.

The above is an introduction to the basic considerations in providing backup for a data base. Without considering the details of particular implementations there are a number of other observations to be made.

Evolution

As well as the difference in size between the files in a user-support system and those, possibly one, in a data base, there is another way of looking at the difference. A user-support system is likely to evolve while a data base is constructed and then slowly changed - but it is fully operational immediately. This means that there is time to develop a backup system as in EMAS, whereas all changes, including the first, to a data base may be considered crucial and there are no unprotected items.

By evolution we mean that the user-support system over the years supports a growing changing number of users with a growing,

changing collection of individual files on a number of computer systems. A large amount of effort is put into smoothing the transitions between machines as the service is upgraded. The upgrading implies that at each change, including the first installation, there is spare capacity. This can be used to provide crude backup facilities until better are developed under operating pressures. Although a user may work in this environment for ten years, none of his files may persist for more than a few weeks - though he would be upset if the compiler for his favoured programming language was destroyed.

The implementation of a data base system will be quite different. The data base may exist before the computer is installed so we actually start with our very first dump and loading the data base is equivalent to a complete recovery. The data base may start at its maximum size and not grow, only change. If it is fully operational immediately after installation then very good backup is required. These two facts mean that for backup design and implementation greater accuracy in forecasting and usage is required. Obviously in practice most systems lie somewhere between these extremes i.e. user-support systems have some large files that persist and data bases are installed in a phased manner if possible and, if successful are liable to grow as extra uses are made of the facilities.

Files and Records

We started this chapter by considering the difference in size between files in a user-support system and a data base. It is also worth considering a comparison between files and records. In general

records are small therefore even if a large number are changed, a dump of these does not strain system resources. We must dump (i.e. copy) changes in this way to guarantee no loss of information. We do n ot or cannot copy the whole data base either because only a small fraction has been changed or because it is too big. Two ways to improve on this would be:

1) to have the data base structured so that over some suitable period e.g. a day, all the changes will be in an area which can be dumped and recovered without interfering with the rest of the data base.

2) to have the data base structured so that logical sub-units are small enough to be dumped completely if the percentage changed rises above a pre-determined level. If it is possible to do dumps in parallel then the more heavily-used parts of the data base may be very well protected and they can be recovered quickly with minimum searching and rebuilding.

We now consider the files in a user-support system again and look at possible variations in the simple schemes outlined above. There may be users who have files that they treat like small data bases. These users might ask for journal-type backup. Large text files which have small amounts of editing done can be backed up by storing the editing commands. If file indices or directories are backed up then journals of these changes will give complete backup for directories. These require a more responsive system than the EMAS one described in Chapter 3. If we also draw into this argument

the EMAS system of partitioning users such that for each group all their files are on the same device, then we have some general guides for backup systems for user-support and data base systems.

Conclusion

Our conclusion is therefore that the distinction between user-support and data base systems is somewhat artificial and that from a backup point of view they should be approached in the way outlined below. We use the term data base to include user-support filing systems in the following suggestions.

1) Divide the data base into units which are as separate as possible both logically and physically. The aim being to have the logical and physical boundaries coincident. If this is achieved then these units can be managed independently. This means
   a) that the likelihood of total loss is less
   b) a total dump need not be done, instead each unit can have a periodic dump depending on the rate of change and the periods for different units can be different
   c) any recovery should be faster and cleaner as there will be less searching to do.
2) Provide facilities for incremental and periodic dumps and journals (all as described as above). If this can be done along with 1) then it is at least possible for users and management to work towards the level of backup and recovery they require in their particular environment.

In the design of the next implementation of backup and recovery on EMAS (Chapter 10) we explore the possibilities of this sort of integration. If it can be cone successfully then it can be used to tackle the backup and recovery problems for computer utilities and networks of computers. A geographically distributed data base is an obvious example of 1). We describe briefly the problems of networks and distributed systems in Chapter 6.

We have assumed throughout this chapter the capacity to handle dumps. Very restricted systems may have more fundamental problems

a) lack of input/output capacity to handle satisfactorily dump and recovery

b) lack of on-line storage to keep satisfactory records for generating dumps

c) lack of space to keep references to off-line material.

Summary

Wilkes (53) has classed file systems as user-support and data bases. We have examined this division and concluded that, although it may be relevant in extreme cases with very small or extremely large files, it makes more sense to work towards logical and physical partitioning of data bases so that the maximum use can be made of a flexible system of incremental, periodic and journal dumps. In Chapters 10 and 11 we look at a design embodying some of these ideas.

Chapter 6

Other Systems

Introduction

In previous chapters we have looked at backup for one general-purpose multi-access system (EMAS Chapter 3), for a number of similar systems (Chapter 4) and for user-support and data base systems in general (Chapter 5). In Chapter 5 we explored the possibility of providing backup facilities for a system supporting varied activities. In this chapter we consider backup facilities as provided on special-purpose or dedicated systems including those with restricted or exceptional hardware provision. Considering the constraints imposed and the solutions which have been tried should further help in defining the backup primitives required for a multi-purpose system. Also, those areas in which difficulty has been experienced can be identified and solutions evaluated. These difficulties can be classed as being

1) due to the restricted nature of the hardware

2) inherent in the sort of service the system is aiming to provide.

Examples of 1) are lack of channel capacity and lack of on-line storage. An example of 2) is a network of computers.

If we consider the problem of a very large data base or on-line storage system to have been dealt with in the previous chapter, then the problem system for this chapter can be one of the following

1) real-time system

2) network system

3) a system aiming for very high reliability.

Obviously these are not clear-cut divisions and any particular system can fall into one or more of these classes.

Real-time systems

We expect a study of real-time systems to give us insight into solving the problem of maintaining a responsive service while suffering from major processor or storage problems.

The obvious answer is backup hardware. If service must be maintained then a second processor must be available for the occurrence of serious processor faults. Assuming that the second machine is not idle but is doing useful work, then there are three things to consider. Firstly, estimating if machine A is going to be down long enough to make the switch to machine B necessary and if so, deciding how quickly machine B can be taken off its current work which may in itself be valuable. Thirdly, there is the time to recover the new system, now with the B processor, to a defined state. The same considerations apply to the loss of main memory on the A machine. For on-line storage duplication of physical storage is one possibility. Again, if this belongs to A and B machines then loss of A and switching to B implies storing B and loading it with the most up-to-date off-line copies of the A storage. This may take an intolerably long time. One more possibility is for one or both machines to maintain identical data bases on the two storage systems.

This provides permanently available, completely acc urate backup with no recovery processing involved and less reliance on off-line dumps.

For those situations where information loss may mean an elaborate and long recovery procedure, but the time when the system is unavailable is short, then the problem is how to recover as quickly as possible and provide in parallel an increasingly adequate responsive service. This means that dumps must be organised for recovery. Searching many transaction logs for individual items is not good enough so complete dumps must be frequent. If different parts of the data base can be restored in parallel then the recovery can be speeded up. First of all, some sort of directory must be set up so that items can be marked as unavailable but coming, then the service can be reopened.

Networks

A network of linked computers operating on one or more data bases provides two interesting backup questions

1) What new problems does a network pose?

2) Does a network in any way make backup easier so that a network can be chosen deliberately as the best way to handle a particular data base?

Booth (1) considers the general problem of organising data bases in a network. The obvious simple solution is to have files backed up where they are stored. Although they may be accessed and changed by programs from other processors in the network, responsibility

for backup rests with the system running on the computer where the file is stored. If information about a file is held at other processors in the network, then when the file is lost and restored in perhaps an out-of-date form, these other records must also be changed. However a network offers the possibility of storing copies of some files at more than one processor. This saves transmission to provide access and provides a backup copy though care must be taken if copies can be independently updated.

As an extreme example consider a program running on one processor in a network and updating files on storage devices associated with otherprocessors in the network. If this program fails then the recovery operation must know the files involved, recover them at remote locations and control any other programs which were accessing the files. One solution is to allow only retrieval of information from remote files.

If the data base distributed over the network is very large then obviously backing up part of it at each processing node is one way of solving the problem of backup for a very large data base.

Martin (32) has a resumé of all of these ideas as applied to real-time systems. However any evidence to be gleaned from reports of what happens in practice tends to suggest that either very simple solutions are adequate or that anything more complicated is more difficult to implement than a description might imply.

Very reliable systems

For very reliable systems, hardware redundancy and checking mechanisms play a much greater part in keeping a computing system available and this takes them outside the scope of this thesis. A report by Randell (38) describes a number of these systems.

Archive

It is unlikely that any systems in the above classes will have problems providing a separate archive service. For a distributed data base the same questions must be answered for archive as for any other part of the file system. Where is it most convenient to store it? How is it referenced from other processors?

Summary

Backup and recovery for real-time systems and networks have been briefly considered. These systems in general have a more stringent recovery requirement than the general-purpose systems considered earlier. This means either providing extra hardware redundancy or using it if it already exists in a manner to aid recovery. Again the distributed nature of these systems means that a total loss is less likely. That is, they at least, already conform to some of the guidelines we outlined in Chapter 5.

Chapter 7

Hardware

Introduction

In earlier chapters it has been said that the intention is to decouple as much as possible backup and archive design from hardware considerations. However, it is too simple to assume that every system has a disc file and a very large number of magnetic tapes. In this chapter we consider the use of magnetic tapes and discs, their use in the future and the impact new storage technologies will have.

Storage media

Magnetic tape

Over the years magnetic tape has been recorded with 7 tracks or 9 tracks and with densities of 200, 556, 800, 1600 characters per inch. According to Canning (7) densities of 3200 and 6250 bpi are now available with a theoretical upper limit for these phase encoded tapes of around 8000 bpi. Taking 2400 feet as a standard length of tape, then ignoring the effects of inter-block gaps, a current 1600 bpi tape will hold up to 50 Mb with prospects of raising it to 200 Mb. This does not represent a large data base though it is obviously greater than many. However it does represent a large number of records, or transactions at a few hundred bytes each, or a large number of files at a few tens of thousands of bytes each. In terms of previous discussions that means that a few tapes may hold a day's logged

changes to a data base, but for a complete checkpoint dump despite these high recording densities, a large number of tapes may have to be written. Therefore for any particular system we will probably only be interested in the transfer rate of magnetic tape decks in relation to mass dumps. If the pleas for modularity are heeded and it is possible to dump or recover portions of a data base in parallel, then the number of decks will be more important than the data transfer rate of individual decks. Even for large numbers (37) tapes are a desirable backup and archive medium because they are portable and cheap to store. Two further problems are that to be useful, very old tapes must be readable on the currently available decks and they must not have deteriorated such that the error rate is unacceptably high. Two recent studies (22, 37) have investigated the reading of old magnetic tapes i.e. up to ten years old. Their conclusions can be summarised as follows:

1) Tapes do give more trouble as time passes so it would appear wise to use them bearing this in mind.

2) Many errors can be removed by exercising and cleaning the tapes.

3) Both surveys indicate three potential trouble spots:

   a) the first 25 feet of tape

   b) the end of recorded data or of the tape

   c) an area about 1200 feet down a tape.

      This appears to be the area most affected by variations in temperature and humidity.

4) There are fewer errors if tapes are properly stored i.e. under conditions of controlled temperature and humidity (37).

Discs

Here we consider discs as on-line file storage devices and not as extensions of main memory. Although discs have been available for 10 years, it is only recently that they have begun to supersede tapes in all branches of the computing community. So there are a lot of users who are only now facing up to the problem of backup and archive when using random access devices. It has been automatic with tape-based systems. These discs range from removable packs holding 7 Mb to fixed disc files with a capacity of up to 1000 Mb and access times of the order of 100 m secs. Removable packs may have a capacity as large as 200 Mb - a suitable module for a large data base.

Hoagland (27) expects the recording density of production disc files to improve, possible by "two orders of magnitude" in the next decade from the present 2×10^6 bits/in^2 (IBM 3340). However he also states:

"A major advance in the reliability of hardware is urgently needed if the full potential of mass-storage devices is to be realised. Improvements in capacity and access time cannot be at the expense of reliability, because users are now placing and maintaining all their vital records on-line under computer control." This implies backup problems may become more acute. With large capacity removable disc packs, one possibility is to use these for backup and archive

purposes. This might be the case if the total volume of material was growing slowly or changing slowly. These discs cost more than the magnetic tapes and are more bulky so they are less portable and require more space. In general the projected improvements in discs and tapes will not make a great deal of difference to the backup and recovery problem, although the use of large-capacity removable disc packs may add an impetus to the sensible use of modularity in data base design. Any new ideas are going to come from the advent of mass storage systems.

Mass storage

The material in this section is based on Houston (28) which contains references concerning individual devices.

Mass storage refers to devices holding of the order of 125,000 megabytes of information i.e. 200 times as much as the on-line storage capacity of the EMAS 4-75. Their characteristics make them look like large automated tape stores. Access time is greater than ten seconds. To be tolerably efficient the largest possible blocks must be transferred and access basically sequential. Therefore they obviously have potential uses as archive stores replacing tape libraries. Many such devices have been described over the past ten years but there are still few reports of extensive practical use. The market for them is obviously still limited and is likely to remain so as long as greater densities are achieved on tape and disc. If an installation can double its storage capacity without a change in technology then it is not going to experiment with untried devices.

Chapter 8

File System Implementation

Introduction

As mentioned in Chapters 1 and 2 many of the considerations involved in backup and archive systems are also involved in any file system implementation. In this chapter we look at the following topics; integrity, security and privacy of information, file allocation and data compression.

Integrity

For on-line storage i.e. main memory, drums and discs there are exhaustive checks to ensure the validity of information and to detect errors as soon as possible. Tapes used for backup and archive purposes must be subjected to as many checks as is reasonably possible as well. The highest possible level of confidence must be established. In addition to parity and checksum techniques, the tapes should be read after writing. This can be done both on the deck used for writing and on a different deck. For archive information there must be copies on different tapes. This allows checking by comparison. In general the backup dumps will contain at least two copies of a file so duplication at the time of writing is not required. If a system is very reliable there is a danger of over-confidence. We suggest that even if backup material is not required in earnest some of it should be used periodically to check that:

a) the information is readable and correct

b) the on-line references to it are uncorrupted

c) the software accessing it is robust in a changing environment.

Security

The existence of backup information reinforces the security of the file system. However we must also consider the security of backup and archive storage. Obviously a complete installation can be destroyed; if work is eventually to continue or information not to be lost forever, we require copies of archive material at a separate site. Backup copies could also be transferred. This is especially true of large data bases involving many transactions. In the case of an exclusively user-support filing system a lot can be achieved by users working from their own hard copy e.g. card or line printer listings. For data bases therefore we suggest similar. Transactions could be recorded at the terminals where they are entered and journals could be copied and removed from the central installation. All this off-site information must be in a suitable form for rapid use. It should be self-identifying. The one thing which cannot be stored is the knowledge and expertise required to put it back into service. There should however be documentation and matching system software and applications programs.

Privacy

We have been considering information loss caused by hardware malfunction or perhaps faulty software. There is also the possibility

of intentional damage or unauthorised access. A recent bibliography lists over a thousand items on this subject (41). From our point of view are there any additional problems raised by the existence of backup and archive files, especially when they are stored outside the computing installation?

As long as the information is on-line, privacy means controls on user and program access, including possibly cryptography. Cryptography is also possibly relevant if the safe copies are transmitted to distant sites by telephone lines, since these could be tapped. If magnetic tapes are transferred and privacy is considered a problem then an extension of the computer room security is required. This means checks on both personnel and access to storerooms.

File allocation

As the range of available storage devices grows and computers are linked in networks the problem of file allocation may have to be considered more seriously in practical situations. In the past the problem has largely been one of allocating files to main memory and the next level in the storage hierarchy. With the advent of successful large multi-access systems the problem is extended. If there are various types of on-line storage device and information off-line, but considered accessible by users, then there must be a system for deciding which information is to reside on which devices and when it is to be moved. In general this is obviously a very complex problem. The EMAS archive system described in Chapter 3 is an example of

a simple problem and a simple, ruthless solution. Another solution would be to use all available on-line storage space and then tackle the congestion problem, instead of avoiding it. However with the current approach we have learned about user behaviour and the effect of rationing. When there is congestion on the disc file the effects will not be as severe on original users as if they had been allowed unlimited resources. We see backup and archiving contributing, not only to continuity of service, but also to a continuing level of service.

As previously mentioned in Chapter 6, the problem also arises with networks. Here there are the added problems of transmission costs and a file being updated from a number of nodes in a network. Casey (12) reports on a study of a mathematical model of file allocation in a network. There is much work to be done to find adequate practical and theoretical solutions. This would help in deciding how much backup and archive storage can be provided by transmitting between nodes of a network.

Data compression

We discussed data compression in Chapter 2. We mention it again here to point out that it may influence privacy and file allocation. Coding makes data a little less vulnerable to unauthorised access though it cannot stop a determined intruder. If coding for compression is sufficiently effective it may mean that a file allocation strategy can be changed.

Chapter 9

Original EMAS Design

1. Introduction

In Chapter 3 we looked at the file backup and archive services which have been provided for the Edinburgh Multi-Access System (EMAS). That was part of a comparative study of similar systems. Before studying the proposals for a more satisfactory and comprehensive service we look at the original EMAS proposals for file protection. These date from 1968 (49). This means that these proposals were made before any equipment had been installed. Thus there was little experience with the problem of running a system and a user service.

We propose then in the succeeding chapters to use these experiences to present another EMAS design. The three inputs to this design are

1) the following design study
2) experience from a simplified implementation (Chapter 3)
3) experience from other systems (Chapter 4)

It must be noted that in this chapter incremental dump is used in the original EMAS sense of a journal.

2. Information Loss

There are several ways in which a file can be destroyed.

1) The storage unit on which the file is stored is lost or destroyed.

2) The storage unit is damaged in some way e.g. a disc surface is scratched by a read/write head.

3) The file is destroyed or corrupted as a result of a system failure (hardware or software) or sub-system failure.

4) The owner (or a 'friend' !) mistakenly deletes the file or appreciably amends it, without preserving a copy of the original.

Recovery of the file is guaranteed if there is a copy in backup storage. However, if the owner has produced his own second copy in file storage, he may be able to recover this (cases 1, 2 and 3 - possibly, case 4 - certainly). Thus the owner makes duplicate copies in file and/or backup storage, depending on the type or degree of protection he requires. The choice between file storage and backup storage should be made carefully, bearing in mind the following

a) backup storage is more reliable than file storage

b) the owner has greater control over files in file storage than files in backup storage. In particular, he can obtain short-term part-protection by making a second copy in on-line storage. As long as the copy is not moved out to archive he may easily delete it when necessary, and so incur a minimum

of storage "rent". (Protection within a session is discussed in Section 3).

c) files are normally recoverable from backup storage only after medium or system failure (cases 1 to 3), Where the user requires protection from his own mistakes (case 4) he should cater for this by duplicating files in file storage. (See Section 4.5).

3. Protection within a Session

When a system failure occurs it is possible that some user processes will have to be terminated. Even so, where a particular session extends for a lengthy period, or where a user is making considerable changes to a file, there is a need for some 'protection of the session'. Then, should a system failure occur it will not be necessary to repeat the whole session. Again, the user may employ such a facility to protect himself from his own 'mistakes', e.g. where he makes any amendments to a file and then wishes to go back to some previous state.

Consider the operations which might be completed in a typical session. In order, these could be as follows

Log in

Edit

Compile

Edit

Compile

Run

Debug

Run

Log out

The period between each operation is a natural breakpoint, when the user may decide which operation to initiate next. He may also use the period to dump partial results (e.g. edited files) where appropriate i.e. he is able to 'protect the session' at this point if he wishes.

A session protection subsystem is provided to handle these file dumps. When a user wishes to protect a file in this way he issues a command of the form

DUMPT(GEORGE) i.e. make a temporary dump
of the file 'GEORGE'.

The file is dumped to a temporary area preferably on a replaceable random access device e.g. replaceable disc. Later in the session the user may recover the file if he wishes but as soon as he logs out all his temporary dumps are destroyed. (During long sessions the user should be able to destroy temporary dumps without logging out). At log out the user may decide to preserve some of the files which he has dumped into the temporary area; he can have these transferred into his file storage area and/or dumped to the backup storage.

To the user the temporary dumping facility is seen as a part of "backup"; in fact, the protection subsystem will probably exist as a part of the file storage system. The more permanent dumping

facilities are handled by the backup storage system and these are discussed in the next section.

4. Backup Dumping and Recovery

As we have already shown, the user may achieve a degree of file protection by making his own duplicate copies in file storage and by using the temporary dumping facility. However at some stage in the development of his program he may feel that the program file now requires a higher level of protection. It is the function of the backup storage system to provide this protection.

Since there is an enormous range in the "value" of individual files from worthless to absolutely indispensable it is essential to provide protection at a level appropriate to each file. Consequently as the user is the only person who can usefully put a value on his file, he should have the ultimate control over the type of protection his file is to have.

Basically the function of the backup storage system is to maintain duplicate copies of the files for which protection has been requested. These copies should be kept on high-quality replaceable media (e.g. reasonably new magnetic tape) in a location which has strictly-controlled access and which is as free as possible from adverse physical conditions. Various methods of dumping and recovering files are discussed in the following sections.

4.1 Instant and Block Dumps

There are two distinct dumping methods

Instant - In which a file is dumped immediately, either by request or at log out

Block - at set intervals e. g. at a specific time each day a dump is made of a large number of files.

Comparing these two methods we deduce the following

1) Block dumping by itself does not give complete protection. Any file which has been created since the last block dump has no protection; any file which has been changed since the last block dump cannot be restored in its current form.

2) Instant dumps, say on tape, have no set order. Where a particular surface on the fixed disc has been damaged it may be necessary to search several magnetic tapes to recover the files. Block dumping can be carried out in a more orderly fashion with due regard to the subsequent recovery problem.

3) An instant dumping facility requires the permanent dedication of at least on replaceable device; the block dumping facility makes no such demand.

From the first two considerations it would seem that if we are to have complete protection and also speedy recovery both types of dump are required.

4.2 One Type of Block Dump - The Fixed Disc

By far the most crippling failure condition, to the file storage system at least, is the loss of all the information on the fixed disc. Contrast this with the loss of one magnetic tape. The

tape holds a maximum of 20 Mbytes (800 bpi, 2400', blocks of 4096 bytes), and such information is almost certainly irrelevant to the users who are currently logged in; by comparison the fixed disc holds 700 Mbytes, of which part is being used by each of the users who are currently logged in. The disc failure thus destroys all active processes and many of the most frequently used files.

One way of reducing this backup problem is to change the function of the disc within the total hardware configuration. Thus if we use the disc as a temporary buffer for files rather than as a fixed storage area, the destruction of the disc copy of a file is not very serious i.e. when we read in files from archive storage to disc we do not destroy the archive copy and this is still available after the disc failure. However to some extent this solution merely transfers the extensive dumping and recovery problem over to the file storage system. Also it does not solve the problem of quickly restoring files to the disc immediately the device is available again.

Let us now investigate the problem of block-dumping the whole fixed disc. The obvious advantage of this is that we are then able to restore 700 Mbytes of information (somewhat out of date) in optimum time when disc failure occurs. Against this we must weigh the following disadvantages

1. The task of dumping 700 Mbytes of information is formidable. Assuming we dump to magnetic tape, 15 tapes (2400' long, 1600 bpi) are required and the

dumping takes approximately $1\frac{1}{2}$ hours. (It is possible to complete the dump in half the time by employing a dumping routine which can output information to 2 magnetic tapes simultaneously).

2. Assuming we do not apply a read-after-write check the recovery of the disc takes an initial $1\frac{1}{2}$ hours followed by further tape-processing to bring files up-to-date. Presumably the system should not be opened to users until these operations are completed.

3. The underlying assumptions behind the disc-dumping philosophy are that

   a) most of the information on the disc requires a high level of protection

   b) at recovery time it is desirable to reload onto the disc precisely those files which were there prior to the failure. This also assumes that the disc is now available.

   c) total disc failures occur sufficiently frequently to justify such a protection mechanism.

To the extent that these assumptions may not be valid in our environment the disc-dumping method will prove unsuitable.

Our overall conclusion is that a much more selective approach to file dumping and recovery is desirable in our environment. In particular we feel that the system should always be opened at the earliest opportunity to the users unaffected by the disc failure and

that file recovery should then proceed in parallel with normal system operation. Apart from a minority of users we assume that most affected users are not too disturbed at waiting up to 24 hours (if this proves necessary) for recovery of some of their files. And note that this applies only to total disc failure. Where a track or surface is destroyed file recovery is much more rapid though file storage by cylinder does introduce some complication.

4.3 A Scheme using Instant and Block Dumps

We have already suggested that both instant and block dumps are necessary. These should be used as follows

Instant Dumps

A user may explicitly request an instant dump of a file by issuing a command thus

DUMP(GEORGE)

Alternatively he may at any time ask for 'standard protection' for one of his files by means of the command

PROTECT(GEORGE)

This will cause file 'GEORGE' to be dumped at the end of any session in which the file has been amended. A file which is receiving this standard protection may also be dumped during the session by means of the DUMP or DUMPT commands.

Dumping which is provoked by the DUMP or PROTECT command is made immediately to a replaceable storage unit (or to several units) almost certainly magnetic tape; the user may not initiate a further operation on the file until the dumping is complete. Each instant dump

is provided with an identifier and so the file recovery routine does not depend on the existence of a reference table (which could get corrupted). Thus the recovery routine locates files by searching tapes. The backup storage system does however keep a table with an entry for each dump tape, showing the period covered by it. This may be used to speed up the recovery procedure in the situation where a user is able to indicate roughly when his file was dumped.

Instant dump tapes are retained for some period greater than the interval between block dumps as a cheap extra precaution and are then recirculated. It is necessary to recover files from the instant dump tapes in the following circumstances

1) After a system failure involving destruction of protected files. It is normally necessary to refer to the last block dump and all instant dumps since then.

2) Where the user (or perhaps a subsystem) has inadvertantly destroyed or corrupted his file in file storage. As we have already suggested file recovery should be allowed here only in exceptional circumstances. Reference may be made to any of the instant dump tapes or a block dump tape.

BLOCK DUMPS

A block dump of files is made at some suitable interval. Weekly may be suitable if the load on the system is low at weekends. A block dump acts as a consolidation of all dumps since the last block dump,

so a suitable saving in the number of tapes involved and the recovery search time can decide what the interval should be. All files which are receiving standard protection and which have been amended since the last block dump are dumped. In addition a user may explicitly request that a file be included in the dump by issuing a command as follows

DUMPB(GEORGE)

This initiates both an instant dump of file 'GEORGE' and the copying of the file to a block dump buffer. When the block dump is eventually made the buffer is first dumped, followed by the copies of the files receiving standard protection. (The instant dump is made merely as protection against loss or corruption of the block dump buffer). When the block dump is made it is possible that some of the files which are to be protected are not currently on the disc; these must obviously be read in from archive initially. Alternatively the file storage system may ensure that either any files which are to be dumped in the next block dump are not moved to archive in the meantime, or as the time for the block dump approaches there is a gradual build-up of the necessary files in the on-line storage.

At the completion of a block dump the dump tapes are added to a pool of such tapes. It is this pool of tapes which forms the bulk of the backup storage library; by comparison, the instant dump tapes serve a somewhat transitory function and so are quickly superseded and withdrawn. A file is recovered from the pool by first searching the pool directory. Each protected file has an entry in this directory,

containing details of all dumps of this file which are currently in the pool. In particular, the appropriate storage unit identifiers are specified so it is unnecessary to search several tapes to locate a file, unlike the procedure for instant dump tapes.

Consider now some examples of how the instant and block dumps are used in recovering files. We assume that instant dumps are retained for a fortnight and that block dumps are made once each week. Figure 3 shows the situation at some random instant in week 'S'. With the notation shown, block dump 'q' occurs at the end of week 'Q'. The instant dumps are divided into two parts, X- containing all dumps made after the last block dump, Y- the remainder.

We now describe recovery procedures appropriate to the following situations

1. Loss of a single protected file

Initially the pool directory is interrogated to discover whether there are any copies of the file available. If there are the file owner is given details, such as when the dumps were made and asked to select a suitable copy. The appropriate storage unit is then brought from the backup library and a copy of the file is made. Should there be no copy available then

either the file is a very old one and all backup copies have been deleted (with the owner's consent). No recovery is possible.

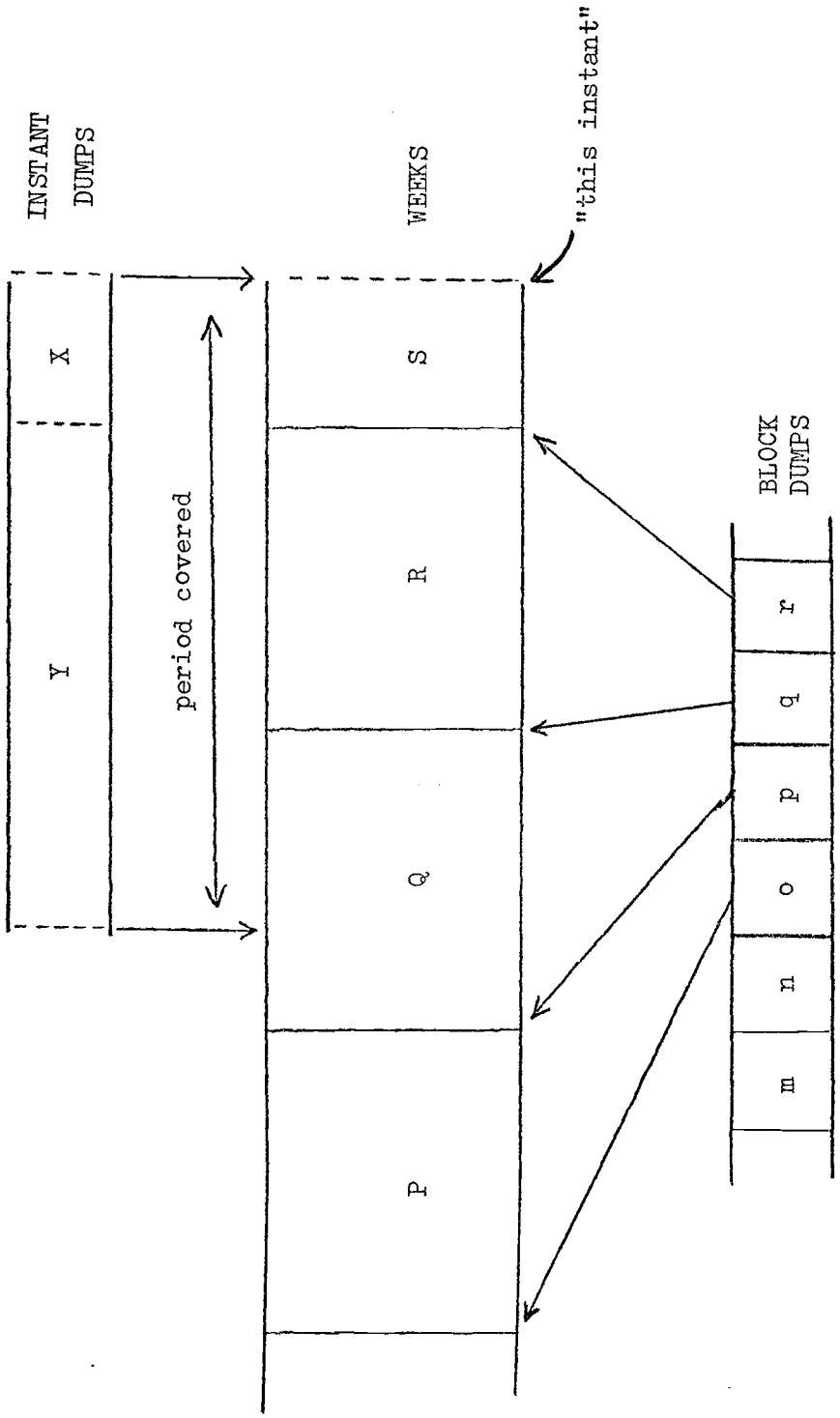

INSTANT DUMPS
X
Y
period covered
WEEKS
"this instant"
S
R
Q
P
BLOCK DUMPS
r
q
p
o
n
m

Figure 3.

or the file has been given protection only since the last block dump. If the user can indicate when an instant dump of the file was made i.e. during the past few days, then the appropriate storage unit may be brought from the backup library and a copy of the file made.

2. Loss of on-line storage containing many protected files

There are two main types of file which need protection and which we may expect to have in the on-line storage

a) files which are being constantly modified e.g. users' program or data files

b) files which may change rather infrequently e.g. some library packages.

When we come to recover such files we will normally find that nearly all the group a) files will be found in the most recent dumps. Thus the 'r' block dump tapes and 'X' instant dump tapes (as in Figure 3) can be used to recover a large number of files in a relatively short period. By contrast the most recent dumps of the library packages may be spread out over a large number of past block dumps and recovery of these could be a lengthy procedure. To obviate such an intolerable situation, it is sensible to make use of the DUMPB command i.e. an explicit request for the inclusion of a file in the next block dump. The most important packages could be explicitly block dumped together say every fortnight or month.

Recovery of on-line files proceeds thus

Assuming the system is fully operational, the file storage system provides a 'recovery problem' for the backup system. This 'problem' consists essentially of a list of protected files which are to be recovered; a recovery priority may also be provided with each entry in the list.

The backup recovery routine normally searches dumps in the following order

Section 'X' of the instant dumps (the last dump tape i.e. the one still on the machine, is searched first and so on).

'r' block dump tapes

Other block dump tapes - in an order compatible with the file recovery priorities.

Files are passed over to the file storage system as they are recovered.

3. Loss of off-line storage containing many protected files e.g. the loss of a magnetic tape

Almost inevitably the recovery priority here will be much lower than in the last case. However the general method of recovering files is very similar- the file storage system sets up a 'problem' for the recovery routine and dumps are searched as before.

Situation where high-priority recovery are needed are

a) where the system has got into a position such that it

it can continue only after a particular file has been brought into on-line storage.

Assuming that the file storage copy is unavailable and that there are no duplicates in file storage, high priority recovery is needed.

b) where a user has logged in and the file storage system is unable to locate the off-line storage unit containing the files required by the user.

4. Corruption of protected files by the user

In the situation where the system manager is willing to countenance the use of the backup system to rescue a user from his own folly, the full recovery facilities are available. The operations are similar to those outlined for the case 1 situation above, 'loss of a single protected file', except that now we may expect to use part 'Y' of the instant dumps in addition to any other dumps. In all other recovery situations but this one, the user will normally wish to have the latest copy of his file; each instant dump of a file is thus made redundant by the following dump and so on. However where a user has been making drastic changes to his file over a period of a week or a fortnight he may find himself in a position where he must go back to a state earlier than the previous dump. In this case the instant dumps can furnish a copy of the file as it was at the completion of each session throughout the previous fortnight. It is for just this purpose that instant dumps are retained for an extended period - strictly otherwise, the instant dumps could be destroyed as soon as a

block dump has been completed (e.g. we could retain only part 'X' in Figure 3).

To discourage misuse of this facility it may prove necessary to make file recovery in this situation both expensive and inconvenient e.g. the recovery operation could be left until a housekeeping session and the file released to the user 24 hours after the request has been made.

4.4 Incremental Dumps

When a large file is to be protected it is inconvenient to have to dump the whole file each time a small change is made to it.

For record-oriented files dumps may be made of transaction journals recording the individual changes rather than dumping the complete file. We call these incremental dumps.

As far as the backup storage system is concerned there is no distinction between whole-file dumps and incremental dumps i.e. there is only one unit of protection. Further it is not the function of the backup system to process the whole-file dump + incremental dumps to recreate a current copy of the file. This is left to the user or a subsystem (see Section 4.6).

Since the backup system makes no distinction between whole-file and incremental dumps we conclude the following

1) The instructions DUMP, PROTECT, DUMPT and DUMPB can be used to protect incremental dumps.

2) Where the user requires protection from his own mistakes he must not depend on recovery of incremental dumps from backup storage. This implies that every file in backup storage whether incremental or whole-file is a second copy or at least an earlier version of some file which exists in file storage.

Consider now how a user can obtain protection for a large file by making explicit dumping requests. Suppose that the file, called 'GEORGE', has changes made to it each day. The user may decide to have the whole file dumped once each week and have incremental protection between these dumps. The whole file is dumped at an appropriate time in the week by means of the instruction

DUMP(GEORGE)

The user may decide to accumulate the changes to 'GEORGE' in a file called 'INCS'. Thus each time a record in 'GEORGE' is changed the new version of the record is added in to the 'INCS' file. Assuming the user does not require protection within a session then the 'INCS' file may be satisfactorily protected by means of the instruction:

PROTECT(INCS)

Suppose now that there is a system failure in which files are destroyed. Consider the three possibilities

1) Only 'GEORGE' is destroyed

The instand dumps are searched to recover the last dump of

'GEORGE'. The user then processes 'GEORGE' with 'INCS' to recreate the current version of 'GEORGE',

2) Both 'GEORGE' and 'INCS' are destroyed

This case is similar to the last one except that the last dump of 'INCS' is also recovered from the instant dumps. The version of 'GEORGE' finally obtained is the state of the file as it was at the end of the last session.

(This particular failure condition may be removed by ensuring that 'GEORGE' and 'INCS' are kept on separate storage units. If the File System can arrange this it is then unnecessary for the user to protect his 'INCS' file).

3) Only 'INCS' is destroyed

The user has two choices

a) He may force a whole-dump of 'GEORGE' and set up a new 'INCS' file.

b) He may recover the 'INCS' file

Alternative b) is used only when the failure occurs between sessions;

alternative a) may be used at any time.

If the user requires protection within each session he can use the DUMPT facility while long-term protection for 'GEORGE' is obtained by using the command

DUMPB(GEORGE) when appropriate.

4.5 Backup Storage Constraints

In Chapter 3 we gave figures showing the number of files

dumped for backup and archive purposes in the recent lifetime of EMAS. The following guesses were not wildly wrong.

Consider the instant dumps. Assuming 200 users dump 5 files per day and the average file size is 20 Kbytes, then the instant dumps for one day (20 Mbytes) will occupy one magnetic tape.

Consider now the block dumps. Strictly there is no simple correlation between size of block dumps since the two serve different functions. However we can probably assume with some confidence that each block will be smaller than the sum of all instant dumps made since the previous block dump. Accepting the above calculations for the size of instant dumps, and assuming a block dump is made once each week, the size of the block dump may be less than 140 Mbytes or 7 magnetic tapes. An average block dump may take then 100 Mbytes or 5 magnetic tapes though this conclusion must be accepted, of course, only as a very rough guess. If we retain block dumps indefinitely then at the end of one year we may expect to have about 5 thousand Mbytes of information (or 250 tapes) in the backup library.

However, we suggest that users should be encouraged to delete backup files from the library, for these reasons:-

1. The above calculations may be hopelessly optimistic and the library could grow to unmanageable proportions.
2. The 'useful life' of most block dump files expires when the next block dump of the file is made. This is certainly true for library packages which are block dumped regularly to

permit rapid recovery rather than to give extra protection. Such dumps should be deleted at the same rate as the dumps are made.

3. As far as possible the user should be unaware of the protection facilities, except on the occasions when he makes explicit dumping requests or when recovery is needed. (Even these operations can be handled by a subsystem - see Section 4.6). As a result, the user will tend to be ignorant of the exact state of the backup copies of his files i.e. how many copies there are and whether they are serving a protective function or not.

The precise method of deleting files from the backup library should be considered carefully. Even where we provide the user with an explicit 'DELETE' instruction it is probably advisable to build-in an automatic delay of, say, a fortnight before a file is actually deleted. This should help to protect the backup library from system failure and impetuous user behaviour. However, strictly this actual deletion of a file is carried out when an entry for this file is removed from the pool directory; the file continues to exist physically until the relevant storage unit is re-issued and overwritten. This occurs when a large proportion of the files on the unit have been 'deleted' - all 'non-deleted' files are then merged with others onto a fresh storage unit. In the interests of good protection however such reprocessing of backup information should occur as infrequently as possible.

In addition to providing a 'DELETE' facility it is necessary to inform the user periodically as to how many backup copies of his

files are currently in existence. He may then choose to delete some of these.

The following sections were written more with a view to protecting the system and the service from users rather than providing useful services for users. So all mentions of constraints imply a fear that users might abuse and overload the backup system. This could be avoided by imposing strict rationing from the beginning. This is wrong both from the point of providing an adequate user service and of implementing software to make the best use of available resources. We suggest that there should be certain constraints on users with respect to the amount of backup information they may have in existence at any given time.

Thus

a) The system manager should be empowered to specify an overall maximum backup storage area for each user. This could be typically 5 Mbytes, though users who are handling files which are themselves larger than 5 Mbytes obviously require a larger backup storage quota.

b) Backup files are to be thought of as 'second copies' or 'earlier versions' of files which currently exist in file storage and have a function only so long as the user retains the file storage copies. Thus when a user deletes a protected file from file storage the backup system automatically interprets this as a signal to delete all backup copies of this file.

c) Since backup storage is not to be used as a general dumping area, but is provided solely to give high protection to valuable files, it is worth impressing this on users by restricting the number of dumps of any one file which may exist currently in backup storage (excluding instant dumps). Obviously users can get round such restriction by producing a second file storage copy under a different name but this is exactly what we would want them to do! Consider the user who is changing his file, 'GEORGE'. So long as there is one recent version of 'GEORGE' in the backup storage the user is happily protected from system failure i.e. given a system failure, he is able to recover a current (or almost current) version of 'GEORGE'. Where the user wants protection from his own mistakes he may wish to go to a much earlier version of 'GEORGE' and here he is not free to use the backup system. In this case it is the user's responsibility to preserve in file storage earlier versions of 'GEORGE' (under different names) as long as they may be required. Whether or not each version has backup protection is of course a different matter. We conclude that the preservation of many dumps of any one file in backup storage serves no useful purpose and should be forbidden.

4.6 Protection Subsystems

From the way we have described the backup facilities so far it would seem that the user interfaces directly with the backup

system i.e. by means of instructions like DUMP and DUMPB the user effectively has his finger on the dumping trigger. While it is true that the user can obtain protection in this way it is also probable that the majority of users will prefer to have the work carried on 'behind the scenes' by some protection subsystem. The subsystem would carry out any or all of the activities described in the previous sections.

i.e. Make instant or block dumps of a file.

Use the DUMPT facility within a session.

Set up and protect incremental dump files.

Create and delete old versions of files in file storage (protection from the user).

Delete backup storage files.

Recover files when necessary.

Recreate large files using incremental dumps.

Subsystems could be developed as follows

1. General Purpose — The user specifies precisely which files need protecting and what level of protection is required.
2. Special Purpose — Each such subsystem provides file protection at a set level and in a specific manner.

Rather than consider a protection subsystem as some distinct entity it is probably more accurate to say that each file subsystem will contain its own protection facilities. Note that protection here

can include both protection from system failure (via the backup system) and protection from the user (via the file system).

Again we have talked explicitly of user file protection. However each subsystem may automatically protect its own working files during a session by means of the temporary dumping facility. A system failure that interrupted the work of the subsystem might then be resolved without recourse being made to the user - in fact the user need never know that a failure has occurred!

5. Protection of System Information

It is important to distinguish between the various kinds of system information. We can consider the following types

1. Copies of the System

The software needed to restart the system is stored separately. This is protected by keeping several copies, both in file storage and backup storage. It should also be stored on different types of media e.g. magnetic tape and replaceable disc to enable the restart to be carried out even where a serious peripheral hardware failure has occurred.

Since the Recovery Procedure may involve the running of engineers' program, possibly with special test media, we must give due consideration to the needs of the engineer. His information may be protected as follows:

Test programs. There should be several copies of test programs on different kinds of media both in file storage and in backup storage.

Test Media. Again there should be duplicate copies of the test media in file storage and backup storage. However, it is likely that the test media for the more vital peripheral devices such as magnetic tape stations will need a higher level of protection than the test media for devices such as card and paper tape readers.

2. Cumulative system data

Costing information and hardware error statistics are examples of cumulative system data. Such information exists in a file which is updated frequently. In many cases a high level of protection is not absolutely necessary and it is possible to use an out-of-date version without causing serious trouble to the system.

Since such files exist essentially as 'dumping grounds' for system information and are interrogated relatively infrequently, it is desirable that they be generated quickly when required. Thus when restarting the system we should not automatically attempt to recover 'current versions' of these files thus putting an added burden on the Phase 1 Recovery Procedure. Rather, we should create new files wherever necessary and convenient so that the system can be opened quickly; recovery of the previous versions may then be made during the Phase 2 Recovery Procedure (see Section 4.3.4). Using this scheme the costing information, for instance, would be stored in a chain of files; these would be merged together only when necessary e.g. for 'book-keeping' purposes.

3. Subsystems

Whether or not users are able to obtain a useful service depends on the availability of subsystems containing command language interpreters, editors, compilers etc. Thus for the majority of users, the system is opened only after appropriate subsystems are made available. It is desirable therefore to make subsystem restoration (when needed) as rapid as possible. This can be done by protecting with the DUMPB facility as outlined in Section 4. 3.

4. Systems under test

After a system failure, a file of diagnostic information is presented to the system expert. In fact there is a change of ownership; the file ceases to be 'system information' and now belongs to the system expert. Conversely when the system expert provides the system with new versions of the system media, he first sets these up in his own reserved storage area and transfers ownership to the system only as the final step. (This approach establishes a controlled interface between system expert and system, and minimises the potential damage a system expert can cause the working system. The extent to which the expert should be allowed or even able to 'control' the working system must be carefully considered).

Bearing these things in mind we see then that system software under test is not technically part of the system. It belongs to a particular system expert existing, perhaps, as data in one of his

problem memories. How much protection it should have must obviously be decided by the expert himself.

Conclusions

These proposals must be criticised on a number of points. Although it is fair not to consider the associated file system in detail the lack of any reference to protecting the directory information which any file system must contain is a glaring omission. Nor is there any reference to the possible structure information the directories may contain.

The user interface also has drawbacks. There are too many too similar commands and too many hints of restrictions.

However, the view of not tailoring plans strictly to the use of a disc file was on the right lines. If that is carried through successfully then one has both more flexible use of available hardware and a potentially smoother transition to a system with new file devices.

Chapter 10

Proposed Backup and Archive Services for EMAS

Introduction

In previous chapter we have considered the backup of on-line storage in many computing systems. Chapter 3 contains a description of the backup and archiving facilities so far implemented in the Edinburgh Multi-Access System. Some of the initial design work for this system is in Chapter 9. That work was done without any practical experience. In the remaining chapters of this thesis we bring all these aspects together and propose a scheme to provide powerful and flexible facilities in the spirit of Chapter 9, but based on the experience of Chapter 3 and reflecting the knowledge gained in other chapters. It is hoped that this will provide some assistance in both the design and implementation of file system backup for future systems whether or not they resemble EMAS.

User Requirements

We begin by looking at all the improvements to backup and archiving, as currently provided on EMAS (see Chapter 3), that users can reasonably demand. As pointed out in Chapter 3 the file system has proved very reliable, both from hardware and software points of view. This has meant no undue pressure to repair the flaws in the backup and archiving service. These suggested improvements are therefore to satisfy immediate demands and to provide more adequate service at any future date when the loss of on-line information is more severe than we have experienced.

1. The current DAILY dump is done regularly but is too infrequent. The aim should be to reduce the period as much as possible but strive to keep it reasonably constant. Once users are aware of the period that information is at risk they may adapt their working habits. This does not mean that they have to be aware of when a dump is done but that if one is done approximately once an hour then a new file will not exist for much longer without a copy being dumped. This point must be considered if system expansion is planned. An increase in on-line storage will not affect the amount of backup material dumped if processing power remains unchanged. However an increase in processing power could mean that the number of files ready for dumping when the next incremental dump is due could not be handled.

2. A similar criticism applies to the ARCHIVE command which can be used to transfer files to cheap long-term storage. Action should be as rapid as possible, certainly not up to a week after the command. The file may very well be protected with backup copies in existence but the user wants this particular file archived to reclassify it and to regain the right to the on-line storage it occupies.

3. In addition to the above dumps there is a need for a further refinement. Users may wish to preserve particular versions of files. This means that they want a say in deciding the backup copy that is dumped rather than leaving it to the standard system. An example where this might be useful, would be a file being written to regularly over a long period of time, e.g. all day. In this case none of the backup copies are necessarily consistent. The uses and abuses of

this facility and the lifetime of the files it produces will be considered in more detail later.

4. Although the user is not aware of the housekeeping involved in controlling the off-line backup storage he does want control of his own backup and archive files since they are really simply an extension of the file storage as he sees it and he wants similar controls on all of it.

5. This raises the question of recording changes of information about a file rather than information contained in it. In particular, at the moment no record is kept if a user destroys a file, so after a loss of information the file may be restored from a backup dump. Similarly any changes in the permitting of files to other users should be recorded so that any restoration of an earlier state is as accurate as possible.

These are the major changes required. All others are refinements in the implementation of them.

System Improvements

Similarly a number of areas for improvement in the implementation can be identified.

1. Automatic recovery of missing files.

2. If it is possible to speed recovery of both a small number of files and a complete file system quadrant then this should be done.

3. The controlling of on-line storage and automatic archiving should be separated from archiving on request. It should also be automated to a greater degree. With the frequent scanning of the file indices that the above dumping proposals will require there will be ample opportunity to collect information on file storage and usage.

4. The increased activity of dumping programs means that they must be more robust. Currently if a dump program fails the whole dump may have to be repeated from the beginning. This is not satisfactory and a failed dump program should attempt to establish what it has done and what remains to be done. It should then complete the dump with consistent records of what has happened.

5. The problem in (4) is really just one of a class that we have not yet properly tackled. These are the problems that arise from failures, whether hardware or software, while any of the dumping or recovery programs are running and the problems caused by running these programs in parallel with a normal user service. In general we have run dump programs with no user service and not allowed a user to run if his file storage were being restored for him. This obviously does not apply to requests for archive files.

The File System

This section briefly outlines the parts of the EMAS file system relevant to the backup and archiving proposals of this chapter. There are also some details in Chapter 3.

The system has a list of known users. A user has a name and a password. All of this information must be protected. The on-line storage is then allocated in the following manner. Each user has a file index which lists the unique names of his files and contains the addresses of the areas of the disc file which they occupy. A file index is fixed in size. This means that a limit is imposed on the number of on-line files a user can own. Also since the addressed unit of storage is fixed there is a maximum number of such addresses which can be held in the index and therefore a limit on the amount of on-line storage a user may have. Until a user requests otherwise his files are treated as being unprotected. The other material in the index which concerns us is the list of access permissions. A user may permit other users to access his files in various modes. This information is stored in the index. If a file is protected then obviously this information should also be protected.

The backup and archiving systems proposed operate at the same level as the file system. The file system treats a file as an unstructured collection of bytes. Any structure is imposed by a subsystem (39). We do the same. Any library structure is contained in files. There may be links to other users' files, but the backup and archive systems make no attempt to check file content to ensure that library references are correctly maintained. This is a separate problem requiring much more detailed investigation. In the following sections we propose backup and archive facilities to be added to this file system. Since this is an exploratory exercise these facilities

will be added rather than incorporated. This means it can be done with minimum disruption or change to the file system or current backup and archive system.

Backup

The aim of the backup system is to keep copies of the files, those which the user wants protected, on magnetic tape. The aims are that from these tapes any file or files can be restored as rapidly as possible and that they will be the most recent versions available. This means minimising the number of tapes involved and having an efficient scheme to address them. We want to avoid recovering items by searching for them on tape. At the same time even if the whereabouts of the desired files are known it would be inefficient if the average number of files restored from any tape mounted was low.

The first part of the backup service is to periodically do a base dump of the file system. This is a base to work from if all on-line storage is lost. Scanning all the file indices we can find those files marked as protected and dump them if they are not in use and liable to change. This is not to be a frozen state of the file system as in the current weekly dump but simply a compact recording of all protected files. Those that are missed will be caught by the next incremental dump (see below). Note that this use of incremental differs from that in Chapter 9. Once they have been copied then all previous tapes can be considered available for use although in practice they will be kept for some further time as an extra precaution.

The second part of the service is to do incremental dumps. Again by scanning all the indices we can discover all the files created or changed since the last dump, incremental or base. An obvious refinement would be to have a record of only those indices which have been changed. The current daily dump is a primitive version of incremental dumping. We would hope to achieve a period of the order of an hour. Again files which are being changed or may be changed will not be dumped. A dump is a copy of a stable state of a file on on-line storage. If a base dump is done only infrequently, say monthly, then there is a lot of material in the incremental dumps which is not required. So some form of compaction may be desirable to satisfy a demand for speedy recovery of a file. This can be done in two ways: it is really a partial base dump rather than an incremental dump. It is a base dump of active files.

1) Add another flag to the file index so that a file can be separately marked as due for partial base dump.
2) Let the backup system work out from its own records (see below) which files are eligible for a partial base dump.

To satisfy the demands we listed at the beginning of this chapter we must also record incrementally the effect of user commands other than those changing file contents. These are destroying or renaming files, setting or revoking access permissions and revoking a protection request. These cannot be recorded by the backup system scanning the user file indices. Information of this nature must be

communicated to the backup system as well as to the file system. Both from the points of efficient access and implementation and development, we want to leave the file indices alone. These infrequent changes can be recorded in parallel with no great overhead by the backup system. All such changes should be recorded on the dump tapes in case a complete on-line reconstruction has to be done by scanning the tapes. However this should be a very unusual event. The normal use of backup tapes should be to provide some specific files which have been lost. We propose that the backup system should keep a backup index per user. This will contain further details about a user's files. In principle it could all be in the file index. However by splitting it up we hope to leave the file index small and efficiently accessible. At the same time the backup system can decide how much of the other information need be kept on-line. It allows also the flexibility of keeping a list of all backup copies of a file so that if there is a tape failure while attempting to restore the latest copy then the address of the previous one is also available. We see this modularity as aiding the efficient implementation of critical areas like the central file system and as a way to solving the problem of implementation and testing while providing a continuous service.

The effect of this dumping and recording procedure should be that if an on-line protected file is lost a copy can be restored that is completely accurate unless the file has been changed since the last incremental dump. The only 'live' items in the dumps are those with names which exist in the current file indices and which were dumped later than the creation of these.

Once such a system is working then information about times of creation and dumping can be made available to users. But only as information about the state of protection not as another level of file storage. In the following sections we discuss how demands of this sort might be met.

Checkpoint dumping

There are a number of things that the above backup system does not provide for users.

1) If a file is connected in write mode for a long time then it will not be included in the incremental dumps over this period.
2) If a file is changing rapidly, e.g. being used to record events then the incremental dumps made of it may not correspond to desirable points at which to record its state.
3) A restricted file index and restricted on-line storage may mean it is inconvenient for a user to preserve particular file states by making copies within the file system.

To explore the possibility of satisfying these demands we propose another module for the backup system called user checkpoint dumping. This will allow a user to ask for dumps to be made. The effect will be immediate and the original file will remain. Whether the dump is another copy on-line or on magnetic tape is an implementation matter. Essentially we are giving the user access to more file storage, this time controlled by the backup system rather than the

file system. The backup system will control an index on the user's behalf and the user will have commands available to manipulate it. In particular both of the following facilities should be available.

1) All checkpoint copies of a file can have the same name as the file system one and the user only has access to the most recent one.

2) A user can give checkpoint copies unique names. This allows him to restore any particular copy to the file system and do it while the master copy is still named in the file system index.

This is viewed as an adjunct to the backup system so the lifetime is short. If a user wishes to preserve files in the checkpoint dumps then they must either be restored to the file system or transferred to the archive system described below.

Archiving

By archiving we mean the provision of cheap long-term storage and another level in the storage hierarchy. This module serves two quite distinct purposes. Given that the standard EMAS file system can be considered restrictive, the archiving system allows users to nominate files for storage off-line. In this way their file index represents the files they are working with and which can be contained in their allowed disc space. In addition if there is a shortage of on-line file space then the situation can be improved by archiving unused, unprotected files.

We propose to improve this aspects of EMAS by acting immediately on a request for a file to be archived. This means making at least one copy and destroying the original. There may already be one or more copies in the backup dumps. The recording of usage as described in Chapter 3 could be improved. Note should be taken of the amount of free file space and the amount in use by each user. Again the user sees this as another file index controlled on his behalf. The reasons for keeping archive material separate are operational but the distinction is worth making because it is an identifiable use of a file system.

Reloading and Recovering

Reloading and recovering is more fully covered in Chapter 11. By reloading we mean the complete re-building of the on-line file system. This is defined by the list of users. Once this is set up then the backup system can be instructed to find for each user the latest available copy of each file identified as belonging to him and eligible to be listed in his file index, i.e. there was an on-line copy when the file system was lost. Obviously if no record of a file being destroyed has been made off-line a few unwanted files may be reloaded. The aim is therefore to find these files and rebuild each index for a user. This means finding the most recent dump of the backup index made by the backup system. Once this has been reloaded any changes to it which can be found in more recent dumps are repeated. It is then possible to compile a list of all the files to

be reloaded, each with its tape address and the addresses of earlier copies in case there is a tape failure. If a backup index cannot be read then an earlier one must be found and a larger scan of the dumps made to bring it up to date. One could wait to complete this rebuilding before opening the service to users. However, this may take many hours and the system will certainly be capable of supporting a user service and the reloading. It might be desirable initially to limit the users allowed on, either in number or by some classification. We see no reason for any special flags in the file system index. Users can be told that the file system is being rebuilt and they will simply see files reappear in the index. Users could be allowed to query the reloader to find out how far back in the dumps it has reached or which outstanding reloads it has for them. This means that only those users with an explicit query need to be serviced. There is no need for every file index access to check file status in case it is 'still to be reloaded'. For checkpoint and archive it is only necessary to reload the appropriate indices. These may have to be updated by scanning recent dumps again but there is no need to restore any files.

By recovering we mean recovering individual missing files. That is the files are identified as missing by the file system and the backup system is asked to supply copies. The most obvious example of this is when there is an inconsistency in the file indices after a system failure. Two files may be recorded as occupying the same area of on-line storage because the disc copy of an index is not

up-to-date. In this situation the files are destroyed and backup copies sought. Currently this is a search of a line-printer index to the dumps followed by retrieval from the tape. The backup index proposed amounts to keeping this information on-line and providing an information retrieval service based on it.

System Failure

In the present backup and archive systems, system failure while these programs are running does not cause undue trouble. In the case of dumping there are no on-line records to be inconsistent. We can simply start again. While reloading or recovering only the current file can be inconsistent so it can be destroyed and its restoration repeated. For archiving a little more care is required since files are destroyed. First the archive index is dumped, then the files copied to tape (two tapes in fact). Only after this is the index changed. Finally the files are destroyed. Any failures before the destroy sequence can be dealt with by starting again. A failure while destroying means that the list of files to be destroyed must be regenerated from the index, if it is safe, otherwise from the self-identifying files on the tapes. This can then be used to repeat the destruction of those files still in existence. There have been very few failures while these programs were being run.

In the proposed system with considerably more activity, both programs running and indices being updated, the chance of disruptive system failure is greater. As pointed out earlier, what we are proposing is a number of file systems run in parallel. Therefore

in the same way that the on-line file system is checked for consistency, the backup, checkpoint and archive indices must be checked as being accurate after a system failure. This is because although a file may have been copied to tape the change to the appropriate index site on disc may not have taken place. Therefore a consistency check means finding in the index the last recorded tape address and checking if any further additions have been made to the tape, or the next tape in the sequence. If so, then a record of them must be added to the index. If reloading must be restarted then the list of files can be reconstructed and the remaining missing ones identified by checking the file system indices.

Summary

In this chapter we have reviewed the defects in the current backup and archive system. Having in addition briefly summarised the EMAS file system we proposed how to improve backup and archiving in this environment. For backup these changes were:

a) base and incremental dumps

b) record changes involving destruction, permissions renaming and protection

c) keep an on-line index of dumped files.

In addition we proposed user-requested checkpoint dumping. The changes to the archive system involve simply separating the cheap storage aspect from that of policing disc space. We then considered the problems of reloading the file system or recovering files from these new dumps. Finally we considered how to deal with keeping

the many proposed indices consistent should the system fail while they are being altered.

Chapter 11

Proposed Implementation

Introduction

In Chapter 3 we gave details of the current backup implementation. Although the results have been satisfactory the implementation leaves a lot to be desired. We propose that all of the necessary functions as suggested in Chapter 10 be taken over by a backup process. This chapter describes the organisation of this process and a phased implementation schedule such that

1) the current scheme and the new one can co-exist

2) new facilities can be tested

3) the current scheme can be discontinued

4) new facilities can be made available.

A full description of an EMAS process and the part played by Director, the paged part of the supervisor can be found in the EMAS reports (33, 39, 43, 50). For our purposes suffice to say that Director provides the file system for user processes. User processes are either executive or normal. This terminology follows Shelness et al (43). Executive processes perform non time-critical supervisor functions such as unit record device I/O. They have the same level of privilege as processes in the resident supervisor i.e. the ability to communicate with any process whether resident or virtual. Normal processes communicate only with their Director processes. Director processes have the same level of privilege as executive processes.

We propose that the backup process be an executive process. This means it will have files controlled by its Director process as part of the File System. The backup process should be different in the following respects. As a user of the system it has a name and is assigned to one quadrant of the file system (see Chapter 3 and Rees (39)). However in order to acquire file space it should have a file index in each file quadrant at any time accessible to it. It will use this file space to maintain file indices for dumped and archived files belonging to users who have their files in that quadrant. In effect this means that the file system is being partitioned such that Director provides the more critical facilities to the user and the backup process handles the less critical. This allows phased, parallel implementations with minimum disruption to existing sections.

These indices are to be used to record changes to dumps and file system indices. They must be organised to aid recovery of a small number of files and to reconstruct on-line storage. They must themselves be backed up though they must not be indispensable.

In the remaining sections of this chapter we discuss the points which have to be considered to implement this scheme satisfactorily.

Indices

For the purposes of discussion we will use the term file system and ignore the quadrant organisation. The proposal is that for each user the backup process will maintain three indices:

1) a backup index

2) a checkpoint index

3) an archive index

These indices control access to a user's files which exist in the backup, checkpoint and archive file systems. Backup and checkpoint files are copies of files which exist in the file system. Archive files do not exist in the file system. These indices will be organised in file space owned by the backup process. They cannot exactly mirror file system indices. It must be possible using these indices to access the files whether they are on on-line disc storage or on magnetic tape. If more than one copy is referenced - as further backup - then more space will be required. The entries in a backup index are related to those in the current file system index for the same user. Therefore unless many versions of one file are referenced the space required is bounded. However, for checkpoint and archive indices there is no relationship. The checkpoint index can be controlled by imposing a limit on the number of copies made or space occupied by this form of dumping. The archive index will grow but a limit may not be necessary as growth will be slow.

If all indices were held on average to one page like the file system index then the total of indices, i.e. four per user, would occupy 2% of the available file space. In addition the backup, checkpoint and archive indices could be held in the file space allocated to the backup process by the file system. By comparison with the file system indices these file system indices will be referenced infrequently.

Only 50% of files have been backed up in the past. Although users may have more archive files than on-line files the on-line ones are what they are using so the archive index will not be frequently required. The use of checkpoint dumps will be even less widespread. Most people will not require it. It must be implemented so that it is not used unnecessarily. Checkpoint indices therefore need not be held for all users. They can be created as required and since they simply come out of the backup process's file space they are not wasting space since it is only claimed as required.

The backup process will maintain these indices and change them, but for requests for information will pass the appropriate index to the user where it can be interrogated as a file by subsystem or user commands.

For access to the file system indices the backup process will use a Director service which will make the index available and control synchronisation with the user.

Dumping

Files have to be dumped, i.e. copied, either on request from a user, or because the backup process has detected by searching that a dump is due. In the case of an archive dump the original file must also be destroyed. The backup process must control the dump devices whether these be magnetic tapes or areas of on-line disc storage used as buffers.

The dumps made after a scan of the file system indices are:

1) base backup dump

2) incremental backup dump

3) archive dump of unused material.

The dumps made after a request are:

1) checkpoint dump of a file

2) archive dump of a file.

Base backup dump

This is done under control of the backup process so it must first request tape decks and tapes. In principle its action then is simple, gain access to each user's file system index, record those files to be dumped, copy them to tape and record the address of this latest backup copy in the backup index. Since this is the start of a new backup sequence the current backup indices addressing the existing backup dumps can be dumped to tape first of all. The base dump then starts the building of a new backup index. It should not be necessary to suspend user service to do this dump. Any file which the user is changing will be ignored. A copy will either be in the previous dump, in which case a reference can be left in the new backup index or it will be dumped in the first incremental dump when it is free. Both of these dumps may exist. The same facts apply in considering whether to write duplicate tapes. Only files created or changed since the last incremental dump will not have an off-line copy on tape. This dump will grow with increased on-line storage but not so much with an increase in processing capacity. If it is likely to take too long, then

it can be split and a base dump done of those users' files partitioned into one of the file system quadrants. These dumps can then be done on separate days.

Incremental backup dump

Again the backup process is in control. The incremental dump will contain copies of files created or changed since the last incremental dump. In this case a marker must be reset in the file system index so that any further change will mark the file as eligible for the next incremental dump. The aim is to do this dump with a period considerably less than the current once a day. Where the eligible files are detected by scanning the file system index,the shortest period would be achieved by starting another scan as soon as one finished. This could be slightly improved if Director supplied a list of those indices containing the incremental dump marker set. Director can easily record this information without accessing the indices. It would save the backup process accessing all the indices. Probably considerably less than 10% of indices would then be involved.

Another alternative would be for the backup process to be notified to take action when a changed file ceases to be in use. However, this raises the prospect of the backup process not being in complete control. If it cannot do the dumps it must queue the requests. If it cannot queue the requests then it must eventually scan the indices.

If it is possible to achieve a very short period then a large number of tapes may be produced and it may be sensible to do a partial base dump at the end of the day. This dump would contain the latest

copies of all files created or changed since the last base dump, complete or partial. The incremental dump tapes intervening would then be freed. This partial base dump is of course the current daily dump.

Archive dump of unused material

If the files which are to be copies to off-line archive storage are identified because there is a need to create more free space on-line then the backup process on its periodic index scans can both identify this need and the files to be dumped. If the decision to dump is taken, as now, simply on the basis of usage over some externally determined period then the backup process has to be used to identify and dump the relevant files. Like the previous dumps the backup process must arrange to have tapes available. There is no need to buffer the files on some other on-line storage site. Once the files are safely on tape the backup process can gain access to the user indices and destroy the original files. There will be a copy of the file in the most recent base dump. This means that should the archive tape be corrupted, it can be reconstructed from material on backup tapes. This will , however, eventually be destroyed. Rather than write a second tape as currently it is more sensible to have a utility program to copy archive tapes when convenient. If the original is corrupted then the archive indices can be adjusted to address the copy tapes and the utility used to make more copies.

Checkpoint dump

Dealing with checkpoint dumps is exactly the same as having files printed on a line-printer. The backup process takes a copy of the file and dumps it when magnetic tape is available. If a tape is available for incremental dumps then it can be put on that, otherwise it can be held with any further requests until a tape is available. If checkpoint dumps are to be held for a number of days or weeks then they must be copied to separate tapes to allow the incremental tapes to be used again. This can be done from on-line storage to tape or incremental tape to checkpoint tape depending on the number of files involved. If there are too many files taking up too much valuable on-line space then the buffering must be cut and the tidying up done tape to tape. However, if this is the case then it probably means the facility is being abused in the EMAS environment. The dump must be recorded in the checkpoint index but nothing is done to the file system index. If the dump has been done first to disc and then to tape the index entry must have the address changed. This is also true for changing tape identifiers.

Requested archive dump

This is similar to a checkpoint dump. However the entry in the file system index must be removed once a copy of the file has been made. The original file may be retained by the backup process so that two copies are in existence. If the file does not already exist in backup storage then it can be added to the next incremental or base dump so that two off-line copies exist as soon as possible. These backup copies are not required once the current archive tape has been copied.

Other dumps

If the backup tapes are to record all changes to the file system index so that accurate reconstruction is not dependent on the contents of the backup indices but can be done by scanning the tapes, then extra files must be dumped containing details of files destroyed and permissions changed. These can be put out in the incremental dumps. They will be superseded by base and partial base dumps. They will be files belonging to the backup process. This is only for identification on tape. They are not required on-line as files, just long enough to record the relevant information, then they can be destroyed.

Recovery

The situations in which files must be recovered from tape are as follows:

1) A file is requested from archive storage.
2) A checkpoint copy of a file is requested from checkpoint storage.
3) The file system determines that some files have been corrupted in on-line storage and requests that if they are protected the latest available backup copies be restored. When the backup process receives these requests it can identify the copies required from the backup indices and issue an ordered list of tapes, read the files and pass them to the owner. The backup index or a buffer will contain up-to-date information about the file. In fact this information may be in the file index if it is only the file storage which has been corrupted.

4) The file system must be restored.. This is obviously a decision taken by an administrator.

The first two cases can be dealt with as now. After the tape address of the requested file has been found in the appropriate index a request for the tape is issued. All outstanding requests for that tape are ordered and the files read as the tape is scanned.

The third case is simply an automated version of the same. The requests are generated by the system rather than users. If we keep references to more than one, either all or some fixed number, backup copy of a file then we can recover from tape failure while reading a file and give the user the most recently available. If it is a base dump read which fails then the latest copy in the previous incremental dumps will be the same. Had the file been changed since the incremental dump immediately before the base dump then it would have been included in the incremental dump after the base dump. If the request list control file is lost then the backup process would be unable to complete the required recovery. Therefore the contents should be available to the system administration so that the requests can be repeated if the packup process is known to have been in trouble, or the files are detected as missing and there is no sign of a recovery in progress. It is obviously desirable both for the backup process to display what it is doing and record it, and for it to allow interrogation by the operations staff or administration.

In the case of complete on-line storage recovery, since the back-up process runs as a process and uses the normal file system services,

there must be a preliminary stage before it starts recovery. This preliminary stage consists of establishing an empty file system. This consists of the file system services provided by Director, a list of user names and passwords, space for their file system indices and details of the available on-line storage. Once the backup process has been started all it needs are the command for the total recovery and the most recent dump tape. From this it can read a dump of the backup, checkpoint and archive indices. It can then bring these as up-to-date as possible by scanning the rest of this tape looking at the files on it. If any changes in the grouping or partitioning of users onto the on-line storage has taken place then similar changes must be made in the backup indices if they reflect this grouping. The backup process now has access to the list of users who require files restored and from the backup indices the tape addresses of these files. It can thus construct an ordered request file and start asking for tapes. Users can be allowed to run and ask for the names of the files which will be restored. If they create a new file with the same name as one about to be restored then this will prevent its restoration. If the users are partitioned and only one group needs recovering because storage on one device has been corrupted then this requires that the recovery command specify which group. By building a command file and dumping it, and recording recoveries, the backup process can stay in control or be restarted in the event of further failures.

If the users are allowed to run and there are spare tape decks then fresh incremental dumps can start. The opportunity could also be taken to write a base dump as files are restored. This would

release tapes and produce a compacted index. The previous tapes would be kept for some time but only accessed by first reloading the indices which address them.

Use of magnetic tape

The tapes written by the backup process will use the simple format described in Chapter 3. The logical unit of information on the tape is the file. The physical unit is the page as for other storage devices. The file is stored as an image of the on-line version the user sees. The extra information about it is stored as a separate one page file preceding it. This can be considered a file belonging to the backup process. Should the need arise, it can be extended to more than one page. This separates information about the file from information contained in the file.

The handling of magnetic tape devices is done by resident supervisor. The backup process issues positioning and transfer requests and handles replies of success or failure. Many of the problems of magnetic tape are operational and there is little the backup process can do to prevent erroneous labelling and handling malpractices.

Consistency

Since the backup process is organising a file system it should perform consistency checks. These must be performed when the system is started. If it is always done automatically then it will not be missed after a system crash when inconsistencies are likely to occur. The standard file system check will check the backup indices

and any buffer areas, as files, since they are simply files belonging to the backup process. If there are any inconsistencies with disc extents occupied then the indices will have to be rebuilt from the most recent backup tape. In addition the backup process must check its indices against those of the file system in case any messages from the file system have been lost. These could cause trouble if they were to destroy, change the permission of or rename a file. If there is any doubt about the contents of the backup indices they can be rebuilt from the current backup tape. Any entries which then do not match the file system indices can be deleted.

Control

The one unusual element of control required in the backup process is to control the period of incremental dumping. If this is of the order of a number of hours then it may be done by a message from the machine operator. If it is of the order of an hour or less then the process must be able to count the periods and act. If it is in fact implemented by building it into the Volumes process which checks volume labels every ten seconds then it can count using these activations. If not then a similar series of kicks from the supervisor will suffice.

Naming and allocation

When a user's file has been copied to tape then the backup process will have an entry in the user index containing the file name and the tape address. If the file is still on disc then the addresses of the areas

occupied must be maintained. If as is suggested above the initial backup process uses only file storage provided to it by the standard file system then the index entry will refer to an address relative to the start of a backup process file and the actual disc addresses will be those of this file maintained by the file system. Ideally, the backup process should have its own file storage, preferably a physically separate device. This can be used as a buffer and provide copies very quickly rather than waiting till the next incremental dump tape is mounted. Once this is done the organisation can exactly mirror that of the file system. If required this could allow Director direct access, rather than via the backup process. If the development of this separate file system is matched to that of the standard one then it may eventually be suitable to put all the indices together under the control of a separate index processor.

This approach outlined above is suitable for a mixed largely unknown user load on a developing system. For a known, bounded, transaction-type system then the backup system can be integrated more deeply from the beginning with the file system. In this case more accurate calculations can be done earlier and backup transactions included as part of the load the system is designed to handle.

Phased implementation

The implementation of the above proposals can conveniently be done in a number of stages.

1) Develop the mechanisms of the backup process to collect information from the file system indices, copy files to tape, record the new indices and access the new indices.

2) Investigate the frequency with which the proposed dumps can be performed. This can be done by collecting the information and recording it but not writing any tapes.

3) Take over the functions of the current backup and archiving programs. Improve these to the level suggested by 2).

4) Add the recording of destroying, renaming or changing the permission of a file.

5) Add checkpoint dumping. This requires more work at the subsystem level on the commands available to the user and the controls on him. The backup process will already contain all the necessary mechanisms for dumping and index control.

Summary

In this chapter we have described how the proposals of Chapter 10 can be developed and implemented. This should be done by:

1) developing in parallel with the existing system

2) taking over from the existing system

3) adding the new facilities.

Everything should if possible also be done with a view to eventual integration with the file system.

Chapter 12

Conclusions

Introduction

In this thesis we have reviewed the problem of providing backup and archiving facilities for computing systems. We have looked at the problem of providing copies of on-line information so that if there is information loss the best possible recovery can be made. Also cheap off-line storage may include files with no copies on-line. We have considered how to handle these. Related topics in file and data base systems were considered for their influence on backup and archive storage.

In the past lip service has been paid to giving high priority to backup facilities in file system implementation. In practice, even if design has been done, the facilities eventually provided have been on an ad hoc basic - with the notable exception of the Cambridge system. We suggest that the design must be done with the knowledge that more than one implementation will be required and it will initially be given low priority. However, if this is taken into account the first attempts should be more satisfactory than in the past. Also later improved versions will be seen as a logical progression providing better facilities.

EMAS

In particular we have studied the work done for the Edinburgh Multi-Access System. We have proposed how this might be improved.

The backup and archive facilities are described in Chapter 3. They have been adequate thanks to:

a) the reliability of on-line storage

b) the structure of the system and the checking done both of which have helped to minimise losses

c) no shortage of resources.

The proposals of Chapter 11 should be successfully implemented. There is not sufficient computing power to greatly raise the number of file accesses and changes. However, there is lots of storage capacity to be filled. Therefore base dumps will be bigger. After monitoring the number of accesses to file indices we estimated that even if every one had caused the file to be written to tape this would have involved writing less than two tapes per hour with the system loaded. There may be difficulty in scheduling use of tape decks if we wish to write incremental, checkpoint and archive tapes separately. This will be made more difficult if there is a steady stream of tapes being used to transfer archive material to on-line storage. If these problems are solved there should be sufficient capacity to investigate the problems of data bases as well.

Major areas which require investigation are:

a) the user interface. The user must have adequate commands to use and manipulate the extensions to the indices that we propose.

b) library structures. Good backup and archive facilities must take account of all references to other files, including other users'. This is of course an example of the data base structure problem.

Acknowledgements

I must acknowledge the advice and encouragement I received from my thesis supervisors, Professor S. Michaelson and Dr. D.J. Rees. Dr. A. McKendrick of the Edinburgh Regional Computing Centre provided much advice and information while the current EMAS backup and archive facilities were being developed. Thanks are due to Mrs. Margaret Murray for typing the draft of this thesis. Chapter 3 is based on a paper submitted to the British Computer Society for publication. Chapter 9 is based on work done with J.A. Welford of ICL during the Edinburgh Multi-Access Project.

Bibliography and References

1. Booth, G.M. (1972). The use of distributed data bases in information networks. Proceedings of the first international conference on computer communication. S. Winkler (Ed.) Washington, D.C.

2. Burk, J.M. and Schoonover, J. (1972). Computer system maintainability at the Lawrence Livermore Laboratory. AFIPS Conference Proceedings, Vol. 41, Part 1, pp. 263-273.

3. Canning, R.G. (1968). Restart and recovery. EDP Analyzer, Vol. 6, No. 10.

4. Canning, R.G. (1972). Computer security: backup and recovery methods. EDP Analyzer, Vol. 10, No. 1.

5. Canning, R.G. (1972). The debate on data base management. EDP Analyzer, Vol. 10, No. 3.

6. Canning, R.G. (1973). The cautious path to a data base. EDP Analyzer, Vol. 11, No. 6.

7. Canning, R.G. (1973). Long-term retention of data. EDP Analyzer, Vol. 11, No. 7.

8. Canning, R.G. (1973). In your future: Distributed systems ? EDP Analyzer, Vol. 11, No. 8.

9. Canning, R.G. (1973). Protecting valuable data - Part 1. EDP Analyzer, Vol. 11, No. 12.

10. Canning, R.G. (1974). The current status of data management. EDP Analyzer, Vol. 12, No. 2.

11. Canning R.G. (1974). Problem areas in data management. EDP Analyzer, Vol. 12, No. 3.

12. Casey, R.G. (1972). Allocation of copies of a file in an information network. AFIPS Conference Proceedings, Vol. 40, pp. 617-626.

13. Chapin, N. (1969). Common file organisation techniques compared. AFIPS Conference Proceedings, Vol. 35, pp. 413-422.

14. Chu, W.W. (1968). Optimal file allocation in a milticomputer information system. Proceedings of the IFIP congress 1968.

15. Clancy, G.F. (1966). Summary of the Backup and Multilevel Storage Management System. Multics System-Programmers' Manual.

16. Collmeyer, A.J. (1971). Data base management in a multi-access environment. Computer, Vol. 4, No. 6, pp. 36-46.

17. Considine, J.P. and Weiss, A.H. (1969). Establishment and maintenance of a storage hierarchy for an on-line data base under TSS/360. AFIPS Conference Proceedings, Vol. 35, pp. 433-440.

18. Corbato, F.J., Clingen, C.J. and Saltzer, J.H. (1972). Multics - The first seven years. AFIPS Conference Proceedings, Vol. 40, pp. 571-584.

19. Daley, R.C. and Neumann, P.G. (1965). A general purpose file system for secondary storage. AFIPS Conference Proceedings, Vol. 27, pp. 213-229.

20. Farber, D.J. and Heinrich, F.R. (1972). The structure of a distributed computer system - the distributed file system. Proceedings of the first international conference on computer communications. S. Winkler (Ed.). Washington, D.C.

21. Fraser, A.G. (1972). File integrity in a disc-based multi-access system. In Operating Systems Techniques, C.A.R. Hoare and R.H. Perrot (Ed.) London: Academic Press Inc. (London) Ltd.

22. Gentile, R.B. (1973). On the reading of very old magnetic tapes. Datamation, Vol. 19, No. 10.

23. Gilb, T. (1973). Data base software: a sceptical viewpoint and some alternatives. Management Informatics. Vol. 2, No. 5, pp. 227-234.

24. Gosden, J.A. (1971). The conceptual requirements for a management information data bank. Proceedings of the IFIP congress 1971, Vol. 2, pp. 861-865.

25. Gross, W.A. (1972). Ultra-large storage systems using flexible media, past, present and future. AFIPS Conference Proceedings, Vol. 40, pp. 957-968.

26. Harker, J.M. and Chang, H. (1972). Magnetic discs for bulk storage - past and future. AFIPS Conference Proceedings, Vol. 40, pp. 945-956.

27. Hoagland, A.S. (1973). Mass storage: past, present and future. Computer, Vol. 6, No. 8.

28. Houston, G.B. (1973). Trillion bit memories, Datamation. Vol. 19, No. 10.

29. Katcher, A.M. (1971). Efficient utilisation of limited access archival storage in a time-shared environment. Proceedings of the symposium on information storage and retrieval. April 1971. University of Maryland.

30. McLaughlin, R.A. (1972). Building a data base. Datamation, Vol. 18, No. 7.

31. Madnick, S.E. (1969). Design strategies for file systems: a working model. File Organisation, I.A.G. Conference 1968. Amsterdam: Swets and Zeitlinger N.V.

32. Martin, J. (1967). Design of Real-Time Computer Systems. Prentice-Hall, Inc., New Jersey.

33. Millard, G.E., Rees, D.J. and Whitfield, H. (1974). The standard EMAS subsystem. EMAS Report 3. Department of Computer Science, University of Edinburgh.

34. Mitchell, C. and Holmes, J.J.J. (1972). File security without magnetic tapes on a DEC System-10. DECUS EUROPE. Proceedings of eighth seminar.

35. Mulford, J.E. and Ridall, R.K. (1971). Data compression techniques for economic processing of large commercial files. Proceedings of the symposium on information storage and retrieval. April, 1971. University of Maryland.

36. Patterson, A.C. (1972). Data Base hazards. Datamation, Vol. 18, No. 7.

37. Poland, W.B., Prine, G.E. and Jones, T.L. (1973). Archival performance of NASA GFSC digital magnetic tape. AFIPS Conference Proceedings, Vol. 42.

38. Randell, B. (1971). Highly Reliable Computing Systems. Technical Report Number 20. Computing Laboratory, University of Newcastle upon Tyne.

39. Rees, D.J. (1974). The EMAS director. EMAS Report 2. Department of Computer Science, University of Edinburgh.

40. Ruth, S.S. and Kreutzer, P.J. (1972). Data Compression for large business files. Datamation, Vol. 18, No. 9.

41. Scherf, J.A. (1974). Computer and data security: A comprehensive annotated bibliography. MAC TR-122. Massachusetts Institute of Technology.

42. Schubert, R.F. (1972). Basic concepts in data base management systems. Datamation, Vol. 18, No. 7.

43. Shelness, N.H., Stephens, P.D. and Whitfield, H. (1974). The Edinburgh Multi-Access System. Scheduling and Allocation Procedures in the Resident Supervisor . EMAS Report 4. Department of Computer Science, University of Edinburgh.

44. Shneidermann, B. (1973). Optimum data base reorganisation points. C.A.C.M., Vol. 16, No. 6, pp. 362-365.

45. Stern, J.A. (1974). Backup and recovery of on-line information in a computer utility. MAC TR-116. Massachusetts Institute of Technology.

46. Teichroew, D. (1971). An approach to research in file organisation. Proceedings of the symposium on information storage and retrieval. April, 1971. University of Maryland.

47. Thomas, R.E. and Baldwin, J.C. (1972). The Chilton Multi-Access System. Software - Practice and Experience, Vol. 2, No. 4.

48. Watson, R.W. (1968). Time-sharing system design concepts. New York: McGraw-Hill.

49. Welford, J.A. and Wight, A.S. (1968). System Backup. Edinburgh Multi-Access Project. Preliminary Technical Specifications.

50. Whitfield, H. and Wight A.S. (1973). The Edinburgh Multi-Access System. The Computer Journal, Vol. 16, No. 4, pp. 331-346.

51. Wight, A.S. (1974). The EMAS archiving program. EMAS Report 5. Department of Computer Science, University of Edinburgh.

52. Wilkes, M.V. (1972). Time-Sharing Computer Systems, Second Edition, Chapter 8. London: Macdonald & Co. Ltd.

53. Wilkes, M.V. (1972). On preserving the integrity of data bases. The Computer Journal, Vol. 15, No. 3, pp. 191-194.