bcc	title	prefix/class-number.revision
	DISK BACKUP SUBSYSTEM	DKBK/S-26

checked	authors	approval date	revision date
checked	Bo Lewendal	1/15/70	
approved		classification: specification	
		distribution: company private	pages: 21

ABSTRACT and CONTENTS

This document specifies the Disk Backup Subsystem for the Model I.

Introduction

This document is a preliminary description of the disk backup subsystem to be used on the Model I. The aim of the subsystem is to insure that any and all information on the disk which may be of some importance be safe-guarded from system malfunctions and some user errors.

The specification discusses the matters of when, how, and where all this gets done. The information contained herein has evolved from a discussion of the disk backup system held between Butler Lampson, Larry Barnes, and myself.

Main Objectives

There are several things one should expect and would like in a backup system. They are as follows:

1) Most importantly, the ability to recover the entire contents of disk storage in the event of a major disaster. For this purpose one needs to have a complete dump of disk files on tape, and the tape files must be up to date (as far as possible).

2) The ability to recover the latest incarnation of a file from tape in case that file has been damaged by the owner or, worse, by the system.

These two capabilities obviously only require that one copy of each file (the latest copy) be saved on tape. It turns out that it is easier to keep around all non-historic versions of a file instead of just the latest, thereby conveniently providing the third capability:

3) The ability of the user to retrieve a specific version of a file specified by its dump time. Of course, a reasonable limit must be placed on how far back in time separate incarnations of a file may exist.

4) The availability of an archives system allowing the user to permanently store a specific incarnation of a file. It should be easy to retrieve such files,

but it might be slow and relatively expensive. The Archive system is intended for use as long time storage of inactive files and as backup for important non-changing system files.

The Backup subsystem will have capabilities 1 to 3 and the Archives subsystem will also have capability 4. Insofar as feasable, the two subsystems should utilize common algorithms and tables and be in some way compatible.

Both the Backup system and the Archives system will be part of the standard utility rather than the basic system. In the event that more than one utility will exist, then both of these subsystems should reside in a part common to all the utilities so that no conflicts or incompatibilities will accrue.

Specification of Backup Subsystem

Only the Backup subsystem will be described in detail here since the Archives subsystem is not really specified yet and will not be implemented for a while. However, the Archives subsystem will have much in common with the Backup subsystem as described here.

The essential aspects of the Backup subsystem are as follows and in brief:

1) Dump procedure: The utility maintains a queue of MIBs in which files have been modified. At periodic intervals, the dump program will look at this queue and dump onto tape all those files of each MIB which have been modified since the last dump time. The dump program will also be able to dump all files, whether modified or not, in order to provide protection against system malfunction.

2) Recovery procedure: To recover all files one need only load the dump tapes in reverse chronological order, ignoring the file on tape if it is already on the disk. To recover a specified file one must search the dump directory on the disk to find out on which tape the file in question is

located. Then one merely scans that tape to find and load the file onto the disk.

3) Format of dump tapes: Each dump tape will be a single-volume file with OS standard labels. User files will take up an integral number of records, with each record containing 2048 words. The first part of each record contains a header, the rest, file data. The header contains a record number, the user number/ disk address of the file's owner, the file name, and the access word and lock list.

4) Directory of dumped files: The master backup directory contains an index page and pages of file names and dump locations, sorted by user number. There is one index word per data page which contains the user number of the user whose files begin on that page.

A detailed discussion of the above aspects of the Backup subsystem follows.

Dump Procedure

The Backup dump consists of three levels: a monthly dump of the entire disk file area, a weekly dump of files modified during the week, and the incremental dumps. The monthly dumps are referred to as "monthly" only for convenience. The actual time period may be different, but will be on the order of a month. The same applies to the weekly dumps. The whole monthly cycle will now be described.

At the beginning of each cycle a complete disk dump will be made. To do this, the dump program simply scans some list of all users, and for each user, it scans the user's MIB and dumps onto tape all files. After all of a user's files have been dumped, the dump program updates the "last dump time" word in the MIB header. As the dump program is dumping user files it is also producing a master dump directory to reflect the contents of the dump tapes. The monthly dump will require at most 15 tapes per fully loaded disk unit since each tape holds 10 million words and each disk unit holds 150 million words.

The incremental dump is designed to provide short term backup of modified files. If the user or the system clobbers such a file the user can recover it again by

making a request to the Backup subsystem.

If a file is dumped while it is being modified garbage will result. Therefore it is necessary to define what is meant by "having been modified" so that the dump program will not dump a file which is being modified. Here are three definitions (in the order of ease of implementation) of when a file has been modified:

1) When the user informs the utility
2) When the user informs the monitor
3) When the file is no longer being modified.

The first definition is the one which will be adopted for the Backup subsystem. When a user process signs off or becomes dormant, the utility will add the user's MIB to the dump queue. If, however, the process becomes active and modifies a file which is in the process of being dumped, then garbage will result. But, this will be taken care of by the utility's adding the MIB to the dump queue when the process signs off or becomes dormant again. Obviously a better method needs to be found.

The dump program is driven by the Backup dump queue, which resides in a special file. For each MIB in the queue it scans all the files in the MIB. Each file modified since the time specified by the "last dump time" word in the

MIB header is written on tape and added to the incremental dump directory. After all files in this MIB have been dumped, the "last dump time" is updated and the MIB is removed from the dump queue.

This incremental dump procedure may be interrupted and restarted as needed. When the procedure is started, the current incremental tape is mounted. If the reel has been partially filled, the dump program scans to the end, backs over the last data record, reads past it, checks the incremental directory for agreement, and then proceeds. For a new reel, the following merging procedure is invoked.

At intervals, or when a new reel is to be mounted, the current incremental dump tape is dismounted. Then the incremental directory is sorted and merged with the master directory. Thereafter, when the new master directory has been written out on a special tape, the incremental and old master directories may be deleted.

At approximately weekly intervals, an intermediate dump will be performed in order to cut down on the number of incremental tapes. One method, and the simplest, is to dump all files changed since the last intermediate dump. This procedure, however, has the serious drawback that a file which has been accidentally deleted since the pre-

vious intermediate dump will not appear on the current intermediate dump even if it appears on the incremental dumps for that period. One method of avoiding this drawback is to make the intermediate dumps by merging the incremental dumps.

A reasonable compromise, which we intend to use, is to use the first of the above procedures and also keep the current set of incremental tapes around until the next intermediate dump. This requires two sets of incremental dump tapes. Two sets of monthly dump tapes will also be kept so that we can guarantee backup of files for a period of at least one monthly cycle, and more, depending upon where in the monthly cycle we are.

Recovery Procedure

To recover a specified file, the user or the system consults the backup directory to find out which versions of the file are on the backup tapes. Then a request is made to the Backup subsystem to recover the file in question. As soon thereafter as possible, the Backup subsystem mounts the appropriate tape and scans it to the required file and then proceeds to reload it onto the disk.

A question arises as to whether we should recover the original unique name of a file. It is proposed that such should be the case if the file still exists in the appropriate MIB. If, however, the file does not exist, by deletion or by system malfunction, then the system will create a new unique name for the file.

In the event that it is necessary to completely reload the disk units, the following procedure will be followed:

1) Load the incremental tapes in reverse chronological order, ignoring the file if it is present on the disk already.

2) Continue in the same way with the intermediate (weekly) tapes.

3) Finish with the complete (monthly) tapes.

While these steps are being performed, the users should be informed that file recovery is in progress.

It is estimated that the amount of time for recovering a fully loaded disk unit is three hours.

It would be desirable to have at least two 1600 BPI tape units so that we can overlap rewind/mounting with reading/writing. However, the Backup subsystem will run with only one tape unit.

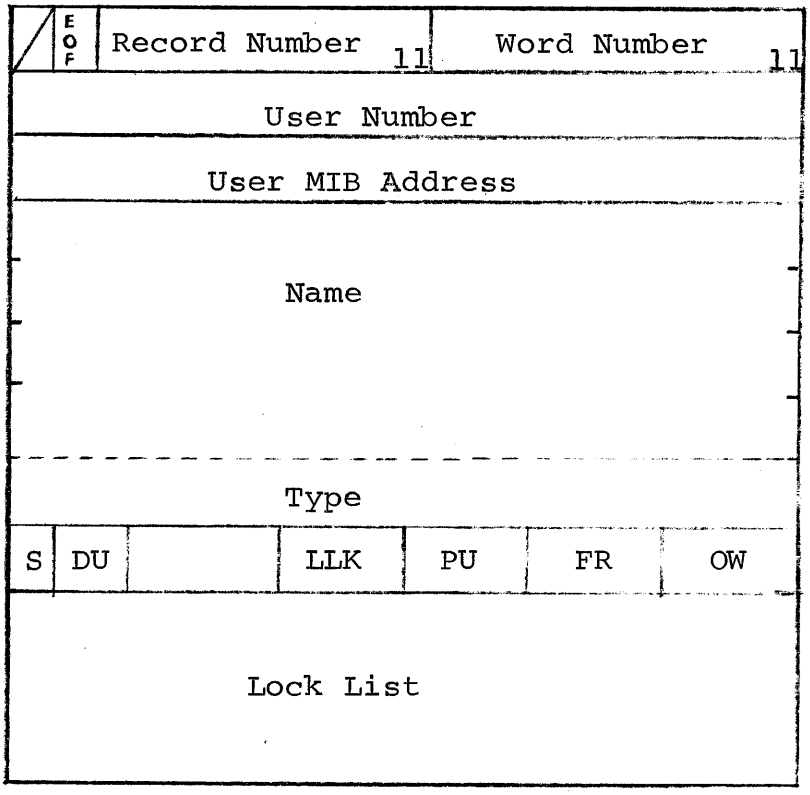
Format of Dump Tapes

A dump tape will be one single-volume file with OS standard labels. User files will take an integral number of records. A record contains 6144 bytes or 2048 words. The first part of each record contains a header. The rest of the record contains the file data.

The header contains a record number, a word number, the user number, the file owner's MIB disk address, the file name, and the access word and lock list. The header is repeated in each record.

If H is the length of the header for a file and R is the record number for the record, then the record contains words $R*(2048-H)$ through $(R+1)*(2048-H)-1$ of the file. The record number is indexed from zero. The last record contains an end-of-file bit and may not use all $(2048-H)$ words. Files less than $(2048-H)$ words are dumped in one record. Note that the header length H is variable since the length of the lock list is variable. The "word number" field of the header indicates how many file data words are in the record, and is useful only for the last record.

The format of the header is as follows:



In addition to the header in each record, a format code and dump time will be stored in the file header label HDR2. The format code is necessary in case we decide to change tape formats at some future time. The dump time in HDR2 is the time at which the tape was first dumped on. The corresponding time at which the dump was halted is stored in the file trailer label EOF2. When a partially filled dump tape is mounted, it is scanned up to the file trailer label which is then overwritten. A new dismount time will be written on a new EOF2 when the tape is dismounted again.

Directory of Dumped Files

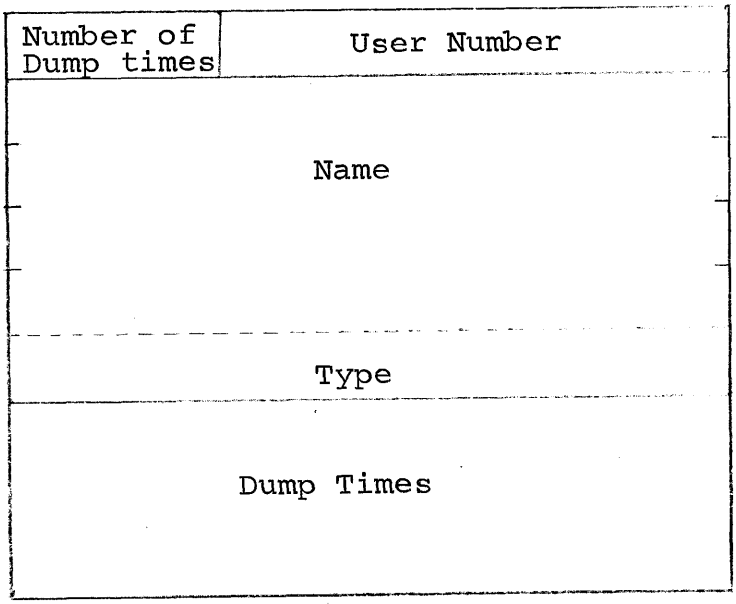
The master backup directory contains an index page and pages of file names and dump locations, sorted by user numbers. Each data page has one index word in the index page which contains the user number of the user whose files begin on that page.

To look up a specified user file in the master directory one scans the index page to find user numbers bracketing the user in question. The corresponding pages are put into the map and the specified file may then be searched for sequentially.

There also exist two identical incremental directories having the same format as the master directory except that there is no index page and that they are not sorted. These directories must be searched before the master directory is searched. They are used for recording dumped files for only a short period of time so that the dump program will not have to go through the time consuming process of modifying the master directory each time a file is dumped onto tape. At intervals, the incremental directories are sorted and merged with the master directory to produce a new master directory, at which time the new master directory will be written on a special tape.

The old master directory and incremental directories may now be destroyed.

The format of a file entry in either the master directory or the incremental directories is as follows:



The two identical incremental directories should be kept on widely separate areas of the disk so that it will be less likely for both directories to be clobbered.

Error Recovery

When it is discovered that the current incremental dump tape does not correspond with one of the incremental tape directories, we follow the following recovery procedure:

1) Compare the two incremental directories; if they are the same go to step 4.

2) Compare each of the incremental directories with the incremental tape; if neither of them agrees with the tape go to step 4.

3) Rewrite the directory which does not agree with the tape with the other good directory; go to step 7.

4) If the tape contains more files than either of the directories, recreate both directories from the tape; go to step 7.

5) If either directory contains more files than the tape and all the files on the tape are in the larger directory, dump those files in the largest directory which are not on the tape and rewrite the smaller directory; go to step 7.

6) Determine when the previous dump tape was last written on, initialize the current dump tape, dump all files in MIBs modified since that time, and recreate the incremental directories.

7) Done. (Send appropriate error messages to Model 30 teletype).

Implementation

The Model I Backup subsystem will exist as a process called M1DUMP. All communication between the user and M1DUMP will be handled by the utility. The utility communicates with M1DUMP via interrupts and the queue files. The dump queue resides in file M1DUMPQ and the retrieve queue resides in file M1RTRVQ.

Since the two queue files may be accessed by both the utility and M1DUMP, we need a mechanism for preventing access to these files by both processes simultaneously. This will be handled in the following way:

Word Ø of each of the two files will be a lock word (LOCKW). If LOCKW = -1 then the file is locked, otherwise the file is not locked. The mechanism for checking and setting LOCKW will be provided by an as yet unspecified monitor call. (XMA will not work when we have two CPUs).

Both M1DUMPQ and M1RTRVQ will be single page files with the first five words of each being:

Ø	LOCKW
1	BP
2	RP
3	WP
4	EP

LOCKW: the lock word as described above

BP: beginning pointer

RP: reader pointer

WP: writer pointer

EP: end pointer

Both queues are in the form of strings with BP, RP, WP, and EP being the required pointers. BP and EP bracket the buffer area which occupies the rest of the file page. WP points to the last entry added to the queue and RP points to the entry which is about to be processed by M1DUMP. The strings have 24-bit bytes.

Format of dump queue entry:

Ø	user number
1	disk address of MIB
2	dump time or -1

Format of retrieve queue entry:

Ø	user number	
1	disk address of MIB	
2	NAME	file name
3		
4		
5		
6	TYPE	
7	dump time or -1	

Each queue will start with one page. When the writer pointer crosses a page boundary, a new page will be added. When the reader pointer crosses a page boundary, the queue will be compressed to get rid of the extra page.

The dump program may be instructed to process M1DUMPQ and M1RTRVQ by sending it an interrupt through the Interrupt Wakeup System.

To send other commands, such as "dump entire disk" and "retrieve entire disk", we connect M1DUMP to a CHIO line.

Error messages and messages to the M30 operator are sent via M30COM to the teletype beside the Model 30.

Problems

We will probably wish to backup dormant processes. However, reloading them circumvents the normal protection mechanisms.

We should use special procedures to dump access keys and user directories.