1081

Bell Laboratories Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **A Tape Back-up System for the Network File Store**

Date- **November 24, 1975**

TM- **75-1273-12**

Other Keywords-

Author	Location	Extension	Charging Case- **39199**
R. J. Elliott	**MH 2C-479**	**2879**	Filing Case- **39199-11**

ABSTRACT

A tape back-up system for the Network File Store, a modified version of UNIX, maintains copies of all disk files on tape. It provides back-up storage from which lost files may be retrieved. The system is semi-automatic and at hourly intervals copies altered disk files to tape, thus providing maximum protection. The dump program does not interfere with normal operation of the File Store system.

Pages Text 9	Other 7	Total 16	
No. Figures 2	No. Tables 0	No. Refs. 1	

E-1932-U (6-73) SEE REVERSE SIDE FOR DISTRIBUTION LIST

BELL TELEPHONE LABORATORIES, INC.

DISTRIBUTION
(REFER GEI 13.9-3)

COMPLETE MEMORANDUM TO

CORRESPONDENCE FILES

OFFICIAL FILE COPY
PLUS ONE COPY FOR
EACH ADDITIONAL FILING
CASE REFERENCED

DATE FILE COPY
(FORM E-1328)

10 REFERENCE COPIES

<AHO,A V
+BARBER,MARK R
<BECKER,R A
BROWN,W STANLEY
+DENES,P B
<FLEISCHER,HERBERT I
<FRASER,A G
<GOLDSTEIN,A JAY
<HAMILTON,PATRICIA
HAMMING,R W
+HANNAY,N B
+JENSEN,O C
<JOHNSON,STEPHEN C
<KEESE,W M
<KERNIGHAN,BRIAN W
<LUDERER,GOTTFRIED W R
+LYCKLAMA,HEINZ
+MARANZANO,JOSEPH F
<MC GILL,ROBERT
MC ILROY,M DOUGLAS
+MCDONALD,H S
MORGAN,S P
<OSSANNA,J F JR
PINSON,ELLIOT N
+PRIM,ROBERT C
<RALEIGH,THOMAS M
TERRY,M E
<THOMSON,M-L
<YAMIN,MRS E E
29 NAMES

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

ACKERMAN,A F
AHRENS,RAINER B
ALCALAY,DAVID
ALLEN,JAMES R
AMITAY,NOACH
AMOSS,JOHN J
ANDREWS,ROBERT J
ARNOLD,GEORGE W
ARNOLD,S L
ARSENAULT,JAMES R

COVER SHEET ONLY TO

ATAL,B S
<BAKER,BRENDA S
BASEIL,RICHARD J
BAUGH,C R
BAYER,DOUGLAS L
BERNSTEIN,LAWRENCE
BERNSTEIN,MS D R
BERRYMAN,EDWARD R
BEYER,JEAN-DAVID
BICKFORD,N B
BILOWOS,RICHARD M
BIRCHALL,R H
BIREN,MRS IRMA B
BLAZIER,S D
BLINN,JAMES C
BLUE,J L
BLUM,MRS MARION
BLY,JOSEPH A
BOCKUS,R J
BODEN,F J
BOURNE,STEPHEN R
BOWERS,J L
BOWYER,L RAY
BOYCE,K J
BOYCE,W M
BOYLE,W S
BRANDT,RICHARD B
BROWN,JAMES W
BROWN,WILLIAM R
BURNETTE,W A
BUTLETT,D L
BYRNE,EDWARD R
CAMPBELL,J H
CAMPBELL,STEPHEN T
<CANADAY,RUDD B
CARDOZA,WAYNE M
CAREY,J H
CASPERS,MRS BARBARA E
CATO,HAROLD E
CAVINESS,JOHN D
CHAFFEE,N F
CHAI,D T
CHAMBERS,J M
CHAMBERS,MRS B C
CHANDRA,R
CHANG,HERBERT Y
CHANG,S-J
CHEN,STEPHEN
CHEN,T L
CHERRY,MS L L
CHIANG,T C
CHODROW,MARK M
CHRISTENSEN,C
CHRIST,C W JR
CLAYTON,D P
CLOUTIER,J E
COBEN,ROBERT M
COCHRAN,MRS A
COHEN,HARVEY
COLE,LOUIS M
COLE,M O
COPP,DAVID H
COSTANTINO,B B

COVER SHEET ONLY TO

CRAGUN,D W
CRUME,LARRY L
CRUM,TED A
CUTLER,C CHAPIN
DAVIS,R D
DE GRAAF,D A
DE LEEUW,C
DEUTSCH,DAVID N
DICKMAN,B N
DIMMICK,JAMES O
DOLOTTA,T A
DONNELLY,MISS MARY M
DOUGHER,J P JR
DOWDEN,D C
DUDLEY,MRS E H
DUFFY,FRANCIS P
DWYER,T J
EDMUNDS,T W
EITELBACH,DAVID L
ELLIOTT,R J
ELY,T C
ERDLE,K W
ESSERMAN,ALAN R
FABISCH,MICHAEL P
FACTOR,R M
FEDER,J
FELDMAN,STUART I
<FELS,ALLEN M
FISCHER,H B
FLANDRENA,R J
FOUGHT,B T
FOUNTOUKIDIS,A
FOWLER,BRUCE R
FOY,J C
FRANK,MISS A J
FRANK,RUDOLPH J
FREEMAN,K GLENN
FROST,H BONNELL
FULTON,ALAN W
FU,CHEN
GABBE,JOHN D
GARCIA,R F
GATES,G W
GAWRON,L J
GAY,FRANCIS A
GEPNER,JAMES R
GEYLING,F T
GIBB,KENNETH R
<GILLETTE,DEAN
GIMPEL,JAMES F
GITHENS,JOHN A
GLASSER,ALAN L
GLUCK,F
<GNANADESIKAN,R
GOETZ,FRANK M
GOLABEK,MISS R
GORMAN,JAMES E
GRAHAM,R L
GREENE,MRS DELTA A
GREENHALGH,H WAIN
GROSS,ARTHUR G
GUERRIERO,JOSEPH R
GUIDI,PIER V

COVER SHEET ONLY TO

GUMMEL,HERMANN K
HACHENBURG,V
HAFER,E H
HAGGERTY,J F
HALE,A L
HALFIN,SHLOMO
HALL,ANDREW D JR
HALL,MILTON S JR
HALL,W G
HANDSPICKER,BRIAN P
HANNAH,JUDY R
<HARKNESS,C J
HARRINGTON,T DORSEY
HARRISON,NEAL T
HARTMANN,ROBERT H
HARTWELL,WALTER T
HARUTA,K
HASZTO,EDWARD D
HAUSE,A D
<HAWKINS,DONALD T
HEATH,SIDNEY F III
HEROLD,JOHN W
HESS,MILTON S
HEWITT,B S
HOEHN,MISS MARIE J
HOLTMAN,JAMES P
HONIG,W L
HOYT,WILLIAM F
HO,MS J
HUANG,VICTOR K L
HUDSON,E T
HUNNICUTT,CHARLES F
HUPKA,MRS FLORENCE
<HYMAN,B
HYMAN,G M
IERLEY,W H
IPPOLITI,O D
IRVINE,M M
<IVIE,EVAN L
JACKOWSKI,D J
JACOBS,H S
JENNESS,R V
<JENSEN,PAUL D
JESSOP,WARREN H
JOHNSON,DAVID T
JONES,M B
<JUDICE,CHARLES N
KACHURAK,JOSEPH J
<KAISER,J F
KANE,J RICHARD
KANE,JOHN M
KAPLAN,F
KAPLAN,M M
KAUFELD,J C JR
KAUFMAN,LARRY S
KAYEL,R G
KELLY,L J
KERTZ,DENIS R
KILLMER,JOHN C JR
KLEINER,RICHARD T
KLEIN,MISS R L
KNOWLTON,KENNETH
KNUDSEN,DONALD B

COVER SHEET ONLY TO

KOETHER,MISS M E
KORNEGAY,R L
KREUTZBERG,JOHN
LAANE,R R
LANDOLINA,W C JR
LANG,A
LEE,DENNIS F
LEGENHAUSEN,S
LEHRMAN,WILLIAM
LEKKOS,A A
LERNER,E M
LESK,MICHAEL E
LESSEK,PETER V
LESTER,THOMAS E
LEVINSON,EDWARD
LICWINKO,J S
LIEBERT,THOMAS A
LINDERMAN,J
LIN,P P
LITOFSKY,BARRY
<LOGAN,MRS V G
LONG,P F
LORENC,ANTHONY
LOYD,D G
LUEBKE,ARTHUR C
LUKACS,M E
LUTZ,KENNETH J
LYONS,T G
MACRI,PHILIP P
MADDEN,MRS D M
MAHLER,G R
MANCUSI,M D
MARSH,MISS M
MARSH,MRS A
MARTIN,R L
MASHEY,JOHN R
MASTROMONACO,FRANK R
MATHEWS,MAX V
MATTAVI,RONALD A
MC CABE,PETER S
<MC CULLOUGH,RICHARD H
MC EOWEN,JAMES R
MC GONEGAL,MISS C A
MC MAHON,L E
MC MILLAN,W F
MC MULLEN,E C
<MC TIGUE,G E
MEINKE,JAMES M
MELCHNER,MELVIN J
MENIST,DAVID B
<MENNINGER,R E
MENON,P R
MERCADANTE,P F
METZ,ROBERT F
MIHALAKIS,J M
MILLER,ALAN H
MILLER,S E
MILLS,MISS ARLINE D
MILNE,D C
MOLINELLI,JOHN J
MOLLENAUER,J F
MORRIS,ROBERT
MORR,PHILLIP L
...
424 TOTAL

+ NAMED BY AUTHOR > CITED AS REFERENCE < REQUESTED BY READER (NAMES WITHOUT PREFIX WERE SELECTED USING THE AUTHOR'S SUBJECT OR ORGANIZATIONAL SPECIFICATION AS GIVEN BELOW)

MERCURY SPECIFICATION.....

COMPLETE MEMO TO:
127-SUP

COVER SHEET TO:
12-DIR 13-DIR 127

COCSFS = COMPUTER FILE SYSTEMS

ELLIOTT,R J
MH 2C479

TM-75-1273-12
TOTAL PAGES 16

TO GET A COMPLETE COPY:

1. BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
2. FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
3. CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.

PLEASE SEND A COMPLETE COPY TO THE ADDRESS SHOWN ON THE OTHER SIDE
NO ENVELOPE WILL BE NEEDED IF YOU SIMPLY STAPLE THIS COVER SHEET TO THE COMPLETE COPY.
IF COPIES ARE NO LONGER AVAILABLE PLEASE FORWARD THIS REQUEST TO THE CORRESPONDENCE FILES.

Bell Laboratories

Subject: **A Tape Back-up System for the Network File Store**
Case- 39199 -- File- 39199-11

date: **November 24, 1975**

from: **R. J. Elliott**

TM: **75-1273-12**

MEMORANDUM FOR FILE

This paper describes a system which maintains a complete copy on tape of all files on the File Store of the Spider Network. One of its principal functions is to provide back-up storage from which lost files can be restored. I will refer to this system throughout the paper as the back-up system. The back-up system is now in use on the File Store which is a variation on the file system of Unix[1].

The back-up system provides disk file protection by maintaining file copies on magnetic tape to be used in case of system failure. There are two sets of programs, dump and restore. The dump programs have an incremental Dumper which copies files to tape, an operator interface and a tape formatter. The restore set consists of user commands, selective file restore and complete disk restore programs.

On reboot, the system command *icheck* automatically scans the file system for corrupted files and restores order; missing files are recovered from tape.

1. Overview

The primary program, an incremental dumper, copies (dumps) disk files to tape and maintains records of these copies on disk. These copies may be recalled when a disk is destroyed or a user wants to retrieve a previous version of a file. File Store initiates the Dumper program hourly.

The basic entity defining each file on disk is an entry called the inode. The inode contains a flag which is set when the file is altered. This flag is reset by Dumper when the file is successfully saved on tape. Dumper also keeps a record of the last two dumps of each file in its inode. Dumper examines the disks for altered files or those which have no previous dump and copies them and their inodes to tape. As Dumper copies the files it maintains on disk a table of contents of the files and their tape positions. It also updates the record of the dumps in the inode. Either of the two versions recorded in the inode may be easily recovered.

When all files are checked, Dumper writes the table of contents on the beginning of the tape. To safely and accurately accomplish this, the File Store system has special tape features. These include tape block addressing which allows accurate positioning of the tape, extended inter-record gaps which enable writing on the beginning of the tape and the ability to read the tape after writing and close without writing an end-of-file marker. Dumper also keeps the number of the last tape block written and the size of the table of contents at the beginning of the tape and uses this data when it restarts. On its subsequent run it seeks to the last block, begins appending more file copies and at the end of the run updates the table of contents.

Dumper uses a set of twenty numbered tapes with initial identifying label blocks. The number in the set may be easily changed if more tapes are needed. The tapes are preformatted to leave space for the table of contents and restart information.

A table on disk has a list of the Dump tapes, the current tape, disks dumped and method of dumping (incremental or complete). Dumper first reads this table and gets the current tape number. It tests that the correct tape is mounted and, if not, it prints a mount request on the console. It then checks all disks listed in the table and dumps the files incrementally or, if instructed, does a dump of all the files. Dumper appends files to a tape until it reaches the end-of-tape, has an error or the table of contents is full. It then continues with the next tape in the cycle.

Even when doing a complete dump of the File System Dumper does not interfere with the operation of the File Store System. If a file is updated while being dumped, Dumper ignores it and the next Dumper run copies it. Since Dumper gets its instructions from the table on disk, changes in disks dumped or method of dumping does not alter the dispatch of Dumper.

By using the interactive program, dmpstat, an operator may log special instructions for Dumper into the table at any time. He need not initiate a special Dumper run. This program also gives status of all Dump tapes.

The restore programs retrieve files either individually upon request or for an entire disk if it has been destroyed. A restore command flags a file for retrieval. It also reallocates deleted files and marks them for retrieval.

File Store automatically dispatches a check program which scans the disk volumes (those given in Dumper's table) for files marked for back-up. Since each file has a record of its dumps, the check program queues them according to their Dump tape and starts a recovery program.

The recovery program looks for the required tapes and prints a request on the console if they are not mounted. From the tape's table of contents it finds the block position of the files queued. It positions the tape and restores the files sequentially as they are stored on tape.

Dumper also copies the inodes (file entries) of each disk to tape. If a disk is completely destroyed, the inodes are copied back to disk with all files marked for back-up. The restore programs automatically recover them. The check program queues these files for back-up and then the recovery program retrieves them.

2. Objectives

The design of the back-up system satisfies the following specifications:

File protection. Each disk file has at least one back-up copy on tape. The incremental dump program, Dumper, is run at short intervals (hourly) during the day to provide maximum protection for recently created or altered files.

Minimal operator interaction. Dumper is semi-automatic and is initiated by the system dispatcher. Dump instructions are on disk and an operator is needed only to mount tapes. Records on disk specify which tape is used and how much of it is written. Dumper does all the checking of tape identities internally. If the proper tape is not mounted it prints a request on the console.

An incremental or full dump of the system can be done while the File Store system is operational.

Reliability. The system is intended to be robust in the face of machine failure and operator error, and has a number of checks to make it reliable. The restore routines skip over corrupted areas on tape and continue a search correctly. Interruption of the dump or restore programs require no change in procedure. They are simply restarted. The beginning of each tape has its dump number, last data block number and a table of contents. Each disk volume contains a record of the tape number and tape position of its last dumps. The tape and disks are not updated until the dump program successfully finishes. These records provide a set of

interlocks which enables Dumper to recopy files which were dumped on a prevous incomplete dump run.

Dump by inode. Each disk volume contains entries (inodes) for each file on the volume. The inode contains complete information about the file including its back-up records. A file does not exist without an inode and is deleted when its inode is deleted. The inodes are numbered and the i-number along with the disk volume number and a sequence number (assigned whenever an inode is reallocated) give each file an unique file identification (ID). The back-up system searches the inodes for files to dump and references all files by their file IDs. All programs read the inodes by tracks (10 blocks). This reduces the number of disk accesses. The scan of the file system is faster using this method than searching each user directory. It also solves the problem of restoring the multiply linked files of the Unix System. The table of contents on the tapes refers to the files by their IDs. This list is more compact and easier to search than if the file names were used. However, since the File Store System no longer allows multiple links the dump by directory would have been more advantageous in restoring lost files. Humans recognize files by names more easily than by a series of numbers.

Efficient tape searches. At the beginning of all the tapes there are records of their contents and Dumper runs. The restore programs find the appropriate file along with its postion on tape in the index. They do not need to search the entire tape. The dump records enable Dumper to continue appending to used tapes with minimal searching for the end of data. Dumper uses a limited number of tapes and recycles them. It redumps threatened files, those dumped on a tape which is being reused or next in the cycle.

Accessible records. A table on disk contains a record of all Dump tapes. The system provides an easy method of obtaining information about those tapes. An interactive program also handles instructions for the dump program.

Extension. A further extension of this dump system is an archive system. The inodes of rarely accessed files would be kept on disk but their data would be on tape. The basic requirements are present and its implementation would not require major upheavals.

3. Structure

The back-up system consists of two sets of programs, dump and restore. Appendix A contains a complete index of the system commands and their calling sequences.

Dump Programs

Mtlab labels and formats magnetic tapes for Dumper.

Dmpstat controls the disks dumped, the method of dumping and the use of a new tape. It communicates this information to Dumper by updating a file on disk which controls Dumper's action. It is interactive and gives information about all the dump tapes.

Dumper is the semi-automatic incremental dump program. It checks for correct tapes and dumps the contents of modified disk files to tape. It reuses a tape by appending data to the end until the tape is full. When it has successfully finished it writes a table of contents on the head of the tape.

Restore Programs

Restor is a user command which marks a file for back-up (retrieval from tape). The user invokes Restor if he overwrites a file or wants to recover an old version. It also marks deleted files, those removed from the user's directory, for recovery. Optionally it lists the inodes of previous versions of the named files.

Rcheck periodically scans the file system for files flagged for back-up. It schedules tapes required and dispatches the recovery program.

Rstat is an interactive program which lists tapes needed for restores.

Recov does the actual restore of files scheduled by Rcheck.

Rest restores inodes after a fatal file system crash.

Restore is a shell sequence used for a complete reconstruction of a disk.

Findi and **Salvi** recreate files which have been deleted, but whose inodes have not been reused. If the latest version of a file is not wanted, these programs are used.

Retriev searches a tape for a file by name and recreates the file on disk. In this case the inode has been reused and there is no record of its last dump on tape. (This program is not yet done).

Mkentry is a system command which puts a deleted file with its old inode into the user's directory. The 'list' option of Restor gives the mode of the deleted file. Mkentry is essentially a Unix link command. After invoking it the Restor command begins normal restore processes which will recover the file.

4. Dumper Support Programs

Dumper cycles through a set of consecutively numbered tapes. It uses a tape until the tape is filled or has an irrecoverable error. It then starts the next tape in the cycle. These "Dump" tapes also have a generation number which is incremented whenever a tape in the cycle is reused. This gives a unique reference number for each dump on a tape.

Dumper requires several special facilities in tape routines which were added to the Unix system[1]. In order to read and find data on irregular tapes, each block on a tape has the block address appended to it. Thus the restore routines may position the tape accurately after a tape failure or after skipping corrupted areas. The subsequent file copies on the tape are not lost.

Dumper writes an index of the tape's contents on the head of the tape after each successful run. To do this it requires preaddressed blocks. These are pairs of blocks, a marker block and a data block which is followed by a long inter-record gap of six inches (Figure 1). Dumper first reads the marker block to position the tape and then writes over the data block. The long inter-record gap protects the next block from being overwritten.

Included in the altered tape routines are options to read after a write and to close a tape without writing an end-of-file. This enables Dumper to write several preaddressed blocks at the beginning of the tape and close without writing the EOF.

A diagram of the dump tape format is given in Figure 2. The File Store command Mtlab formats Dump tapes by writing two identical label blocks followed by pairs of preaddressed blocks. The labels contain the name "DUMP", a tape number, the date labeled and the number of preaddressed blocks which follow. In case one of the label blocks contains garbage the restore programs could still identify the tape from the other. By default there are 30 preaddressed blocks. These suffice for the index to the entire tape. All blocks on the tape contain 512 bytes. A 2400 foot tape reel can hold about 20,000 data blocks. The 'C' Language structures for the tape blocks are in Appendix B.

A record of the last two Dumper runs is in the first two pairs of preaddressed blocks. These control blocks contain the generation number, number of dump runs on this generation, date and time of run, last block written on this tape and the last index block used. The remaining preaddressed blocks have the table of contents or index of files dumped and their position on the tape.

Dumper writes several other types of blocks on tape: inode blocks (a copy of the file entries on disk), file descriptor blocks and data blocks. Except for the actual file data, each type

contains a special octal number which identifies it for the restore routines' use for verification when searching the tapes.

A disk volume descriptor block preceeds its copy of inode blocks on tape. It contains the volume number and the number of inodes.

A file descriptor block precedes the dump of the contents of each file. It has the file identification, path name and a copy of the file's inode.

A record of all tapes in the cycle and instructions for dumper are on disk. For each Dump tape this "generation table" contains the date, generation number, the last tape block written and the disk volumes dumped. See Appendix B for the "C" language structure. The File Store command Dmpstat is an interactive program which manipulates the generation table. It gives the status of Dump tapes and logs instructions for subsequent Dumper runs into the table. Dmpstat controls which tapes to use, volumes dumped or ignored and the method of dumping. Thus the operator may request volumes dumped and tapes changed without altering the dispatch of Dumper itself.

5. Dumper

Dumper is the major program in the back-up system. The File Store dispatcher automatically starts it at hourly intervals. Dumper gets the current tape number from the generation table on disk, reads the tape label and checks that the appropriate tape is mounted. For a new generation tape it begins appending dumped files after the last preaddressed block. If this tape generation was used previously, Dumper seeks to the last position given in the control block and appends the dumped files.

The File Store file system keeps dump information on disk[1]. Dumper refers to the files by their file IDs in its tape index. Flags in the inode identify types of files and their current state. Those flagged as special are file system volumes (e.g. RP disks,tapes) and the temporary files are not dumped by Dumper. The flags also indicate files that have been changed since they were last dumped (and are therefore candidates for dumping), or files to be recovered from tape. The generation numbers of the last two dumps of the file are in the inode. If the last dump is on the current tape the block position on tape is given.

Dumper uses a privileged system call DUMP to open a file. The system call EDUMP closes the file and updates the dump record in the inode with the block position on tape. If the file was written while being dumped EDUMP gives an error on return and Dumper ignores the dump. The next Dumper run will copy the file. On a successful dump, Dumper adds the file ID and block position to its index list. As it fills the index blocks it writes them to a temporary file on disk.

Directories in Unix are just a type of file and are treated as such by Dumper. They contain a list of file names and their IDs (inode number and disk volume number). Whenever a directory is altered, Dumper copies it to tape as if it were a file.

Dumper checks the inodes of each disk volume listed in the generation table for files to dump. It dumps files which have been altered or never dumped and are not special, temporary or flagged for back-up (recovery from tape). If a tape drive is faulty (having many errors) or there are many altered files, the tapes are cycled rapidly and copies of files on the next tapes in the cycle are endangered. Therefore, Dumper also dumps any files whose last dump is on one of the next three tapes in the dump cycle. This assures that each file always has a copy on tape and the copies are not lost when the tapes are rewritten.

After either scanning all the volumes or reaching the end-of-tape marker, Dumper writes the table of contents with the IDs and tape position of all dumped files on the beginning of the tape (the index blocks) and updates the tape control block and generation table on disk with the last position on tape. When an end-of-tape condition has caused this activity, Dumper automatically requests the next tape, redumps the file it was copying and continues the scan. A

dump of a file does not cross tapes.

The super block, a block on each disk volume which has vital information for Unix, has the generation number of its last dump and the last block position on that tape. Dumper updates these at the end of a successful run or when the end of a tape is reached.

Since Dumper puts the tape block position of a file's dump in its inode when it is dumped and does not update the tape control block until the end of a successful dump, Dumper is not affected by abnormal interruption or termination. Those files dumped before the interruption will have a block position greater than that recorded in the tape control block, and the next Dumper run will recopy them.

Whenever Dumper accesses a volume it checks the generation number of the volume's last dump. If it is different from the current one it scans all the inodes for files dumped on that tape and changes the block positions to the generation number. The block position is a positive number and the generation number is negative. If the tape position of any dumped file was greater than the position recorded in the super block, then that Dumper run was incomplete. Dumper zeroes the generation number in the inode and the file will be redumped. This assures copies of files will not be lost over a system disaster or change of tapes. After changing the inodes, Dumper writes the current generation number and a tape position of zero in the super block. It also updates this position when it finishes a run.

Since a volume has a record of its last dump it can be unmounted and Dumper can run without confusing dump records. The block position in the inodes always refers to the generation number recorded in the super block and not necessarily the one Dumper is using.

To protect Dump tapes from being accidentally overwritten by tape commands which use drive 0 as a default, Dumper expects the tapes on drive 6 or 7. It checks both drives for the appropriate tape. If the tape is not mounted it prints a mount request on the console and sleeps for a minute. It retries ten times and then exits.

Dumper copies the generation table and all the inodes of each dumped volume to tape each time it starts a new tape in the cycle. This simplifies the reconstruction of a complete volume. A tape inode block which identifies the volume and has the number of inode blocks precedes the copy of the inode blocks on tape.

To prevent over-allocation of a user's space during a complete restore of a volume, Dumper dumps the inodes of deleted files to tape. If there were no records of deleted files, a restore could not distinguish between deleted and active files and would attempt to restore all of them. When a file is deleted, the File Store file system sets the modified flag in its inode. Dumper copies the inode into a file descriptor block and writes it to tape. A call to Edump turns off the modified flag.

The preaddressed blocks on a tape may be corrupted by a faulty tape handler or if used for the wrong purpose. If a new generation tape has such errors in format, Dumper relabels and reformats the tape.

In addition to an incremental dump, Dumper can do a complete dump of a volume. Dmpstat initiates this mode of operation. When Dumper finishes a complete dump on a volume it changes the mode back to incremental dumping in the generation table.

6. Restore Files

The restore programs backup individual files or reconstruct entire volumes. The file retrieval programs are the same in both cases. These programs and their source code are kept on a disk volume which is frequently copied to a backup disk.. In case that disk is wiped out another is always available.

The user command **Restor** marks a file for back-up. The system routine REST sets the back-up flag in the file's inode. Since the inode has a record of the last two dumps, the user has the option of restoring from either dump. As a special function, Restor reallocates a deleted file and marks it for back-up if its inode has not yet been reused. System changes to the link command allow an entry of a deleted file to be put into a directory. Optionally, Restor prints the file IDs of all previous versions existing in the system and the user can select the one he wishes by invoking Mkentry (see below).

Rcheck searches the inodes of volumes for files flagged for back-up. The File Store dispatches Rcheck at hourly intervals. It checks all the volumes listed in the generation table. Rcheck also takes an argument list of volumes and then searches only those volumes given. It prints tapes needed to restore files and with a 'list' option as an argument it prints the files.

Rcheck sorts the back-up files according to the generation number of their tape copy and writes these lists into disk files with the corresponding dump tape names. It keeps a master list of restore tapes in a restore table on disk. This is a variation of the generation table. If it finds back-up files it initiates the Recov program.

Rstat is an interactive command which interprets the restore table. It prints tapes required for back-ups, volumes which have back-up files or a list of the files themselves.

Recov restores the files queued for back-up by Rcheck. It checks two tape drives for the tapes listed in the restore table. If none are mounted it prints a request for the required tapes on the console, sleeps and retries. After ten tries it exits.

Recov writes a copy of the index list on tape to a temporary file. It checks each index entry against the queue of back-up files for that tape until all files in the queue are found. A file may have several copies on a tape so Recov scans the index in reverse order. Since the index is sequentially ordered by tape position the last copy of the file on the tape will be matched first. The queue of missing files is sequential both by volume number and inode number. Recov searches it for a match to index entries by using binary chops. That is, it examines an entry in the middle of the queue to determine which half its entry is in and then the middle of that half to determine which quarter. It then sequentially searches this quarter of the queue for a match. When Recov finds a match it flags the index entry and removes the file entry from the back-up queue.

The missing files are already sorted by tape position in the index list. Recov reads the processed index list and restores the flagged entries. It searches directly to the block position given in the index list and requires no back and forth movement of the tape.

Recov opens the files with the system call REST. It allows directories to be written, does not alter the modify flag and inhibits all other writes. The system call EDUMP closes the files and does not alter the date the file was modified. When it completes all restores from a tape it removes the tape's missing queue and its name from the restore table. It repeats the recovery process if there are more tapes in the restore table.

7. Disk Volume Restore

If a disk volume is destroyed, **Rest** obtains the volume's inode list from the beginning of the most recent Dump tape and writes it back on disk. It then scans the file descriptor blocks for files from that volume. The inode entries from these descriptor blocks are copied to disk, overwriting the existing entry. This places the latest possible version of the inode on disk. Rest turns off the modify flags and sets the back-up flag in all allocated inodes before copying them to disk. Rcheck and Recov can then restore the entire volume automatically.

Rest has other optional functions. If the generation table is lost it restores it from tape. It restores files on specified volumes by simply reading the file descriptor blocks on tape and restoring those with their last dump on this tape. This method requires a complete search of a tape and is used only if the index on a tape is so corrupted the recovery program cannot read

it.

Restore is a shell sequence which is used as a model for reconstruction of a volume. The system must be unavailable to users during the procedure and it is booted in single user mode. See the Unix manual on boot procedures. It mounts the volume, formats it for the File Store System, restores the inodes, clears the accounts and starts the normal file restore program Rcheck.

8. Restore Deleted Files

If a file has been deleted, its dump records are not available using routine system commands. One needs to know the inode number to find the tape which has its last copy.

Findi lists all inodes on a volume which have the name of the deleted file. **Restor** lists inodes with the same path name that have an existing dump on tape. If the inode has not been reused its dump record is still intact. A system command IPR gives the generation numbers.

The interactive command **Salv** searches Dump tapes for the given file ID. It reads the tape index and lists the block positions of any copies of the file. The user can then specify which copy to restore.

A system command **Mkentry** puts the deleted file into the user's directory. This, followed by a Restor command on the file, initiates the normal restore procedures of Rcheck and Recov. The Restor facility for deleted files automatically uses the Mkentry procedure when the latest version of a file is wanted.

9. Testing and Maintanence

Testing and maintaining the back-up system requires tape and disk scan programs, cross-checks of dumps and a log of I/O errors.

Rdmp scans Dump tapes and prints labels and control blocks. It then becomes interactive and requests the blocks to be scanned and the type of block. **Rdmp** interprets the index, inode and file descriptor blocks. Rdmp also prints any block in octal or decimal.

Dumper keeps a log on disk of all I/O errors or format problems encountered. Those errors are also written on the console.

Chkdmp checks that all normally dumped files have a back-up copy on disk. This is a check on Dumper both as an incremental and a compete dump program. It is essentially a modified Rcheck program. It lists all files by their last dump on a tape. The command Chkstat queries the master list of Dump tapes required for a complete restore and the files queued for each tape.

10. Extension

The back-up system needs a method of restoring deleted files by name. If a file's inode has been reused there is no record on disk of its dump to tape and the programs, Findi and Salv, are useless. However since the full path name of a file is kept in its tape descriptor block, a program could search any Dump tape for the file and then restore it.

The system would be more reliable if there were several back-up copies on tape along with older versions of the files.

A "slow cycle" dumper that saves all files could be easily implemented. It would mark them as saved on the slow cycle and thus reduce the amount of data handled by the incremental Dumper since the back-up copies of rarely changed files would seldom be in danger of being overwritten. The slow cycle also would offer backup storage of several versions of the file system. This feature becomes important as additional disk storage capacity is added to the system.

11. Acknowledgement

I am indebted to A. G. Fraser for his guidance in the design and implementation of the back-up system.

R. J. Elliott

MH-1273-RJE-cs

Att.
References
Appendices A, B
Figures 1, 2

Reference

[1] A. G. Fraser, *Variations on the Unix File System* (unpublished internal memo)

APPENDIX A

Back-up System Commands

A complete writeup of the commands is in the Network File Store reference manual files. They can be listed by executing the command Man.

[1] /etc/mtlab [-key] label number [preadd]

Mtlab labels and formats tape. **Label** contains the type of tape, 'dump' in this case, and **number** of the tape. The optional argument **preadd** contains the number of pairs of preaddressed blocks. Default is 30. **Key** is the tape drive number. Default is 0.

Example: /etc/mtlab -0 dump 10 20

This command labels the tape on drive 0 as 'dump' tape 10 and formats it with 20 pairs of preaddressed blocks.

[2] /etc/dmpstat -key [arg...]

Dmpstat issues commands to Dumper by logging them in the generation table or by obtaining information from the table. The **key** argument controls the action of dmpstat. It lists status, volumes dumped, starts new tapes and designates volumes and method of dumping.

Example: /etc/dmpstat -l

This command lists all Dump tapes, the date started, and number of blocks on the tape.

[3] /usr/dump/dumper

Dumper copies files to magnetic tape.

[4] restor [-key] file ....

Restor marks a file for back-up. The **key** option allows deleted files to be backed up or prints the file IDs of previous versions.

[5] /etc/rje/rcheck [-key] [vol]...

Rcheck scans the named volumes for files flagged for back-up. All disk volumes have a unique number which is in the super block of the volume. A list of the volumes dumped is in the generation table. It has an option which will list the files which are found.

Example: /etc/rje/rcheck -p 57

Rcheck scans volume 57 for files to restore, queues them according to dump tape, lists the tapes and the files and initiates the Recov program.

[6] /etc/rstat [-key] [tape no.]

Rstat lists tapes needed for restore. The **key** options print volumes which have files to be backed-up and the file names.

Example: /etc/rstat -v 15

This command prints on the teletype all volumes which have files to be backed up from Dump tape 15.

[7] /etc/rje/recov [no]

Recov recovers back-up files from tapes. **No** is the tape drive number.

[8] /etc/rje/rest [-key] vol

Rest reconstructs the inode entries on the named disk volume. Depending upon the **key**, Rest restores the generation table or the inodes from a Dump tape. By default all files for the volume are restored.

Example: /etc/rje/rest -0i 52

Rest copies the inodes of volume 52 from the tape mounted on drive 0 to disk.

[9] sh /etc/rje/restore

Restore is a shell sequence for the complete restore of a volume. The system is booted in single user mode. The operator must modify the shell sequence for the volume to be restored. It mounts the uncorrupted file systems so they are recognized by File Store and creates a new file system on the corrupted volume. The new system is then mounted and its inodes restored. It is unmounted and then remounted to get the new root inode into core. The accounting is updated and a restore of files is started. The sequence given below is for volume 57.

```
/etc/mount /dev/rp01
/etc/mount /dev/rp02
echo mkvol; mkvol /dev/rp1 pn 57
/etc/mount /dev/rp1
echo restore -i; /etc/rje/rest -i 57
/etc/umount /dev/rp1
/etc/mount /dev/rp1
echo acheck; acheck -s
/etc/rje/rcheck
```

[10] /etc/rje/findi vol name

Findi searches a volume for an inode with the given name. If a file of the name is found on the volume its i-number and sequence is printed.

[11] /etc/rje/salv vol ino

Salv searches the tape index for the given file ID and prints the tape position if found. It is interactive and restores the file on request.

APPENDIX B

C Structures

[1] Tape Label block

```
struct tlabel
{
        int     t-type;     /* type = label */
        int     t-blk;      /* block no. */
        char    t-tape[8];  /* dump tape */
        int     t-numb;     /* number of tape */
        int     t-date[2];
        int     t-paddr;    /* no. preaddr. blks */
        int     t-spare[10];
};
```

[2] Tape Control block

```
struct control
{
        int blktyp;     /* type = preaddressed */
        int blkn;       /* block number */
        int gen;        /* generation number */
        int idate[2];   /* date begun */
        int endata;     /* last data block */
        int dmp;        /* dump number */
        int idxblk;     /* current index blk */
        int dx;         /* index to next file */
        int ivol;       /* vols. on tape */
        int bdata;      /* beginning of data */
        int frwds[9];   /* free words */
};
```

[3] Tape Index block

```
#define NFILES 84               /* files in index blk */
struct finfo
{                               /* file identifier */
        char    f-seq;          /* sequence number */
        char    f-vol;          /* volume number */
        int     f-ino;          /* inode number */
        int     f-blk;          /* tape position */
};
```

```
struct index
{                                                   /* tape index block */
        int             id-type;        /* preaddressed block */
        int             id-blk;         /* tape block no. */
        int             id-nxt;         /* next file */
        int             id-spare;
        struct          finfo           id-info[NFILES];
};
```

[4] Tape File descriptor block

```
struct fdesc                                        /* precedes data blks */
{
        int             fd-type;        /* type = data */
        int             fd-blk;         /* block number on tape */
        char            fd-minor;       /* minor device */
        char            fd-major;       /* major device */
        int             fd-ino;         /* i-number */
        struct inode    f-inode;        /* file store inode */
        char            fd-nam[40];     /* path name */
};
```

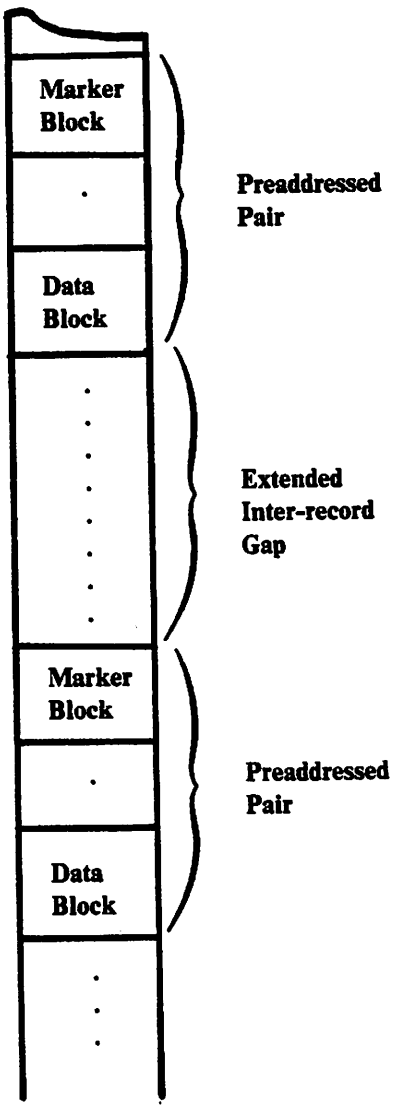
[5] Tape Block Type identifiers

```
#define lblblk 0177560                              /* label block */
#define preblk 0177561                              /* preaddressed block*/
#define inoblk 0177562                              /* inode block */
#define dtablk 0177563                              /* data block */
```

[6] Generation table /* record on disk */

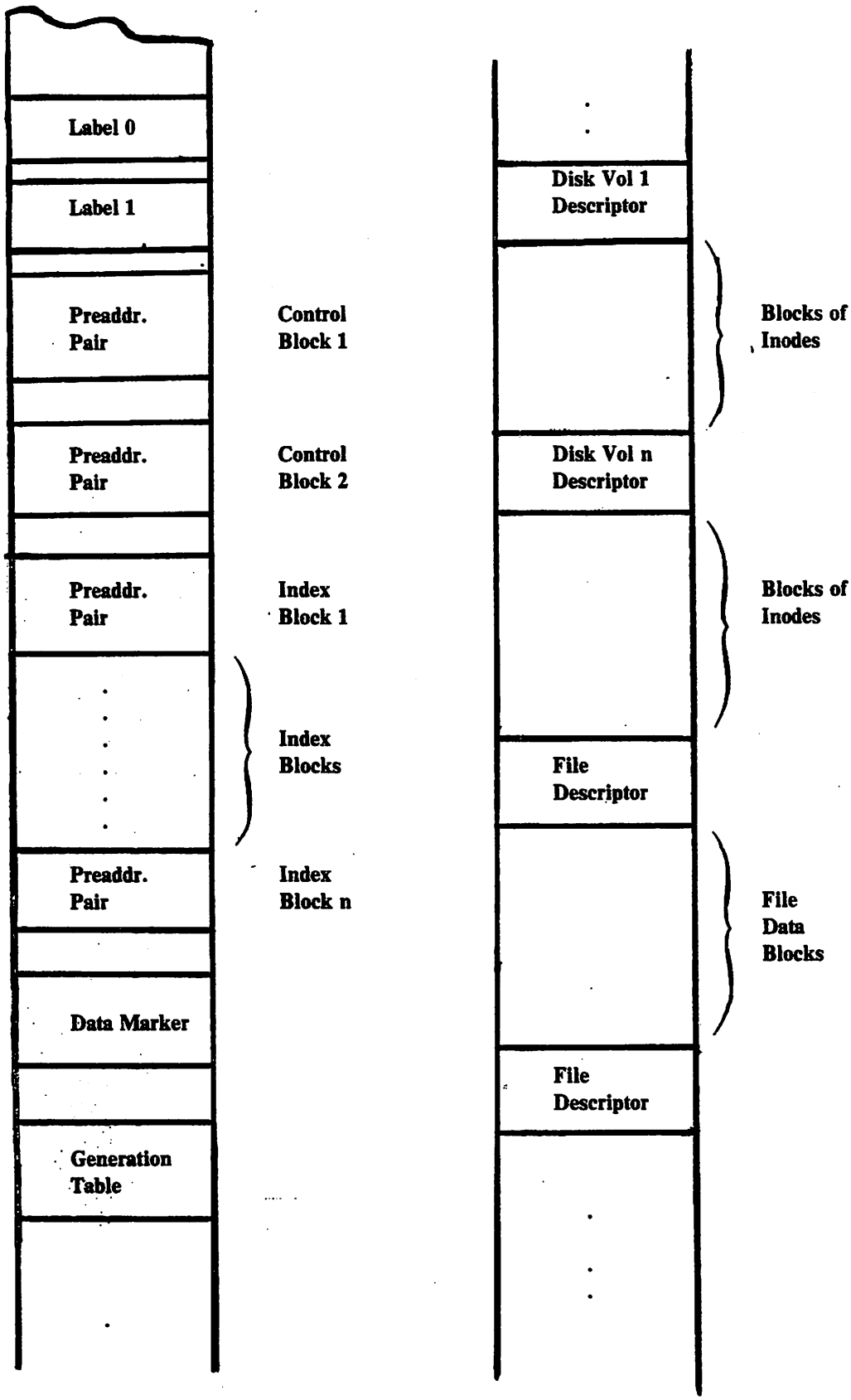
```
#define NTAP n                                      /* number of tapes. */
struct gtap
{
        int             g-numb;         /* generation number */
        int             g-date[2];      /* date started */
        int             g-blk;          /* next data block */
        int             g-vol;          /* bit map of volumes */
};

struct gtble
{
        int             g-cur;          /* current generation */
        int             g-idx;          /* current tape */
        int             g-nxtblk;       /* next data block */
        int             g-dmpa;         /* complete dump vols */
        int             g-dmpi;         /* volumes ignored */
        int             g-spare[11];
                                        /* tape information */
        struct gtap     g-tap[NTAP]
};
```



Preaddressed Tape Blocks

Figure 1



Dump Tape Format

Figure 2