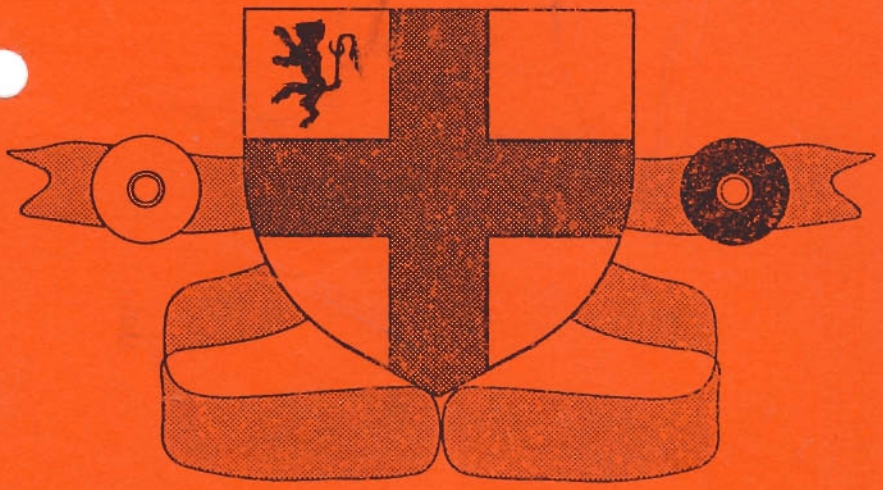
File Recovery System

Restores "lost" OS9 Files and Directories



P.O. Box 733
Maple Valley, WA 98038

PUBLISHED BY BURKE & BURKE
P.O. BOX 733
MAPLE VALLEY, WA 98038

COPYRIGHT NOTICE

LIMITED WARRANTY

Burke & Burke warrants the PRODUCT against defects in material or workmanship for a period of ninety (90) days from the date of purchase by the original owner. This warranty is limited to repair or replacement of PRODUCT which proves defective during this period, at the sole expense and discretion of Burke & Burke. This warranty specifically excludes software defects and defects caused by negligence, abuse, accident, or tampering.

DISCLAIMER

Every effort has been made to ensure the accuracy of this manual and the quality of the PRODUCT it describes. Burke & Burke makes no warranties, whether expressed, statutory, or implied, of any kind whatsoever, as to the merchantability of the PRODUCT or it's fitness for a particular use, except as set forth above as the LIMITED WARRANTY.

Burke & Burke does not assume any liability for any damages, whether special, indirect, or consequential, resulting from the use of the PRODUCT. The PRODUCT is sold on an AS-IS basis.

1.0 Introduction

Thank you for purchasing Burke & Burke's File Recovery System program. The software will help you to recover OS9 disk files that have been "lost".

This section of the manual will introduce you to some of File Recovery System's capabilities. The instructions in the Up and Running section will allow you to start using RECOVER right away.

Chapter 2 provides detailed instructions for each of the programs on the File Recovery System disk.

The educational and technical information in Appendix A will help you make the best possible use of File Recovery System.

1.1 Features Summary

Burke & Burke's File Recovery System (FRS) is a collection of OS9 disk utility programs. There are five programs on the FRS disk:

BA	Sets allocation bitmap bits for a specified Logical Sector Number (LSN) or for a range of LSN's.
BD	Clears allocation bitmap

	bits for a specified LSN or for a range of LSN's.
MV	Move a file's directory entry from one directory to another without copying the file's data.
RECOVER	Recover lost files and directories by rebuilding their directory entries or file descriptor sectors.
ZAP	Erase a file's directory entry without deallocating sectors.

The most important of these programs is RECOVER, the file recovery program. RECOVER works in conjunction with OS9's DCHECK utility to locate and rebuild lost files. Consider these features and advantages of RECOVER:

- Works on any type of OS9 disk, including RAM disks, floppy disks, and hard disks.
- Recovers lost files or directories automatically, placing them in a specified directory.
- Uses existing lost file descriptor sectors where possible.
- Tracks upward through lost directories to reduce file system

corruption.

- Fully disk-based with no memory restrictions.

- Command line options allow RECOVER to display file recovery actions without actually modifying the disk.

1.2 System Requirements

The File Recovery System utilities will run on any Level 1 or Level 2 OS9 system with at least 24K of free memory and one disk drive.

1.3 Up and Running

If you've lost files, it's possible that OS9 has forgotten some of the information used to keep track of where data is stored on your disk. This can happen for a number of reasons; the most common are power failures or program crashes during disk access, and accidental disk modification by defective programs.

You're probably anxious to recover your files, but please read through the manual first. It contains valuable information that will allow you to recover your files faster.

The rest of this section describes the

procedure that you will go through to recover your files. It assumes that you are generally familiar with OS9. If you need a refresher, refer to the Tandy OS9 manuals or any of the excellent thrid-party tutorials.

1.3.1 Where Do You Stand?

The first step in recovering files off of your disk is to make sure that the disk's "file system" is intact. Running the file recovery utilities on a corrupt disk could cause additional corruption.

Every OS9 system is shipped with a utility called DCHECK, which will automatically determine whether or not the file system is intact. If the file system is not intact, it is corrupt. You must use DCHECK and other utilities to place corrupt file systems in a stable state before running the RECOVER utility.

DCHECK creates and modifies several work files as it executes. If you need to recover files from a disk, but are not sure whether or not the disk's file system is intact, you should have DCHECK store its work files on a disk that is known to be intact.

1.3.2 Standing on Solid Ground

DCHECK displays information about corrupt areas of the disk, followed by a summary report. The summary report will say whether the file system is or is not

intact.

If the file system is intact, it is safe to run the RECOVER utility to recover your lost files.

If the file system is not intact, you must use the ZAP and BA utilities to correct the problems reported by DCHECK. Then re-run DCHECK.

This process of running DCHECK, ZAP, and BA must be repeated until the file system is intact.

1.3.3 File Recovery

Once the file system is intact, you can run RECOVER. It's important that no other programs modify the disk that you are recovering between running DCHECK and running RECOVER; disk access by another program could confuse the software, resulting in loss of data.

The RECOVER program uses DCHECK's work files as input, and also creates its own work file. The work files tell RECOVER which sectors on your disk are "lost" (i.e. allocated, but not part of any known file).

RECOVER places any files that it finds in a directory that you specify. It's important that this directory existed at the time that you ran DCHECK; if not, RECOVER will try to recover it, too.

The program makes three passes over the lost sectors on your disk. The first pass recovers directories, the second pass recovers files, and the third pass creates new files from groups of lost sectors.

Each pass can recover data from your disk, but the program advances to the next pass only if no data was recovered in any previous pass.

The RECOVER program will terminate, and will ask you to re-run DCHECK, any time it performs an action that would invalidate the information in the DCHECK work file. For example, when a directory is restored all of the files that it contained are often automatically restored, too. The DCHECK work file indicates that these files are "lost", so RECOVER asks you to re-run DCHECK. This produces an updated work file.

When RECOVER has recovered as much data as it can, it will terminate with the message "Recovery complete."

1.3.4 Renaming Your Files

RECOVER automatically assigns names to recovered files. Sometimes these names are the name of the first OS9 module found in the file, but more often they are derived from the number of some sector in the recovered file.

You will probably want to rename some of the recovered files, and will want to

move them from the file recovery directory to a more appropriate directory. The MV utility on the FRS disk allows you to do this quickly and efficiently.

It is possible that RECOVER will create two or more files with the same name if there are several lost files that contain the same OS9 module. In this case, the DIR command will show all of these files but only the first one can be accessed. You can correct this situation by using the OS9 RENAME command or the MV utility from the FRS diskette.

1.3.5 OS9 Command Lines

You must use the OS9 and FRS utilities in specific ways when recovering lost files from your hard disk. This section shows typical command lines.

The examples in this section assume that you are recovering data from a hard disk /h0, and are using a ram disk /r0 for temporary storage. You can substitute whatever names and device types your system supports.

The command line to run DCHECK when using FRS is:

```
OS9: dcheck -pbmw=/r0 /h0 & w
&xxx
.
.
-xxx
OS9:
```

```
(repair if file system NOT intact)
(run RECOVER if file system intact)
```

The value 'xxx' is the decimal process number of DCHECK. DCHECK uses this number to name its work files. The number may be different every time you run DCHECK. You should make a note of this number so that you can use it when running RECOVER.

The command line to run RECOVER is:

```
OS9: recover /r0/dcheckXX0 /h0
.
.
OS9:
(rerun DCHECK if requested)
```

The value 'XX' is the hexadecimal equivalent of the decimal process number reported most recently by DCHECK. If the hex number has only one digit, the leading digit of 0 must be specified. For example, if xxx=003, XX=03.

Refer to Chapter 2 for more information about RECOVER and the other utilities on the File Recovery System disk.

2.0 Software Reference

This chapter describes how to use each program on the File Recovery System disk.

Each program has a built-in help feature, which can be invoked by using the '-h' or '-?' command line option when starting the program. For example, you can get help for the RECOVER program by typing:

```
OS9: recover -h
```

Built-in help gives you a summary of the program's legal command line options and command syntax. This chapter includes detailed information that is too extensive for built-in help, including special program features and precautions.

These programs are described in alphabetical order:

BA	BD	MV
RECOVER	ZAP	

Each program description includes several types of information, each of which is described under a specific heading. The headings are:

Function: What the program does.

Syntax: What to type into the computer to start the program. Items in [ ]

braces are optional.

Parameters: How to specify the main objects that the program manipulates (e.g. the name of a hard disk that is to be recovered).

Options: How to control the built-in program operation options.

Notes: Detailed description of program operation.

Examples: One or more examples of program command lines, including a discussion of what the example does.

Burke & Burke believes that the information under each heading was correct at the time of printing. If your questions about any of the utility programs are not answered in this manaul, contact Burke & Burke for additional technical support.

BA

Function: Allocate clusters in a device's allocation bit map.

Syntax: ba [opts] <device> <l_list>

Parameters:

device	The name of the device to be accessed.
l_list	A list of the sector numbers (LSN's) to be marked as "used" in the device's allocation bit map.

Options:

-h -?	Built-in help

Notes:

BA sets one or more bits in a device's allocation bit map. The bits to be set are specified as a list of Logical Sector Numbers (LSNs), seperated by spaces. LSNs are specified using hexadecimal (base 16) notation. Each item in the l_list has one of the following formats:

xxxxxx	Set bit for cluster containing specified hexadecimal LSN.
xxxxxx-yyyyyy	Set bits for clusters in specified range.

xxxxxx:nnn	Set bits for clusters that correspond to nnn sectors, starting at specified sector. NOTE: nnn is a base ten number.

The allocation bit map tells OS9 which disk sectors are in use and which sectors are available. Each bit corresponds to one or more disk sectors (also called a cluster of sectors). If a bit is set, all sectors in its cluster are unavailable; a clear bit indicates that all sectors in the cluster are available.

The normal size of a cluster is one sector, but other sizes can be selected via the IT.SAS field of each device's OS9 device descriptor.

OS9 automatically sets bit map bits for one of four reasons:

- The cluster contains a defective disk sector. The set bit keeps OS9 from trying to store data in the defective area.

- The cluster contains part of the OS9 kernel, used to bootstrap the system. The set bit keeps OS9 from erasing the kernel and making the disk unbootable.

- The cluster is being used as part of a file or directory. The set bit keeps OS9 from erasing the file or assigning its sectors to another file.

- The cluster is part of a non-OS9 partition. The set bit The set bit keeps OS9 from storing data in the partition.

You can manually override these automatic operations by using the BA utility.

BA is typically used to repair damaged file systems or to specify a disk defect list.

WARNING: USE EXTREME CAUTION WHEN MODIFYING THE ALLOCATION BIT MAP WITH BA OR BD. IMPROPER USE OF THESE UTILITIES CAN CORRUPT THE OS9 FILE SYSTEM, AND MAY DESTROY DATA STORED ON THE EFFECTED DISK.

Examples:

The standard FORMAT utility automatically marks off defective sectors on your hard drive, but the test used to detect a defective sector is very simple: OS9 tries to read the sector once, and the sector is declared defective if an error occurs. This test may not detect marginally bad sectors. If you know that certain disk sectors are defective, the BA utility can be used to mark them off manually. For example, if sectors $000427, $000428, and $001372 are defective:

```
OS9: ba /h0 427:2 1372
```

Note that sector numbers are always specified in hexadecimal, and that sector counts are always specified base 10.

BD

Function: Deallocate clusters in device's allocation bit map.

Syntax: bd [opts] <device> <l_list>

Parameters:

device — The name of the device to be accessed.

l_list — A list of the sector numbers (LSN's) to be marked as "free" in the device's allocation bit map.

Options:

-h
-? — Built-in help

Notes:

BD clears one or more bits in a device's allocation bit map. The bits to be cleared are specified as a list of Logical Sector Numbers (LSNs), seperated by spaces. LSNs are specified using hexadecimal (base 16) notation. Each item in the l_list has one of the following formats:

xxxxxx — Clear bit for cluster containing specified hexadecimal LSN.

xxxxxx-yyyyyy — Clear bits for clusters

	in specified range.
xxxxxx:nnn	Clear bits for clusters that correspond to nnn sectors, starting at specified sector. NOTE: nnn is a base ten number.

The allocation bit map tells OS9 which disk sectors are in use and which sectors are available. Each bit corresponds to one or more disk sectors (also called a cluster of sectors). If a bit is set, all sectors in its cluster are unavailable; a clear bit indicates that all sectors in the cluster are available.

The normal size of a cluster is one sector, but other sizes can be selected via the IT.SAS field of each device's OS9 device descriptor.

OS9 automatically clears bit map bits when deleting files or directories. You can manually override this automatic operation by using the BD utility.

BD is typically used to repair damaged file systems.

WARNING: USE EXTREME CAUTION WHEN MODIFYING THE ALLOCATION BIT MAP WITH BA OR BD. IMPROPER USE OF THESE UTILITIES CAN CORRUPT THE OS9 FILE SYSTEM, AND MAY DESTROY DATA STORED ON THE EFFECTED DISK.

Examples:

Sometimes a power failure or system crash can create "lost" sectors on a hard or floppy drive. These sectors are marked as allocated in the bit map, but are not part of any file or directory (see also BA for other reasons that a sector might be allocated). You can use the BD utility to release these "lost" sectors so that they can be used to store files. For example, if sectors $001234-$00139B are "lost", you can return them to available status with the command:

```
OS9: bd /h0 1234-139B
```

Note that sector numbers are always specified in hexadecimal, and that sector counts are always specified base 10.

MV

Function: Move file or directory to another location in OS9's tree-structured file system.

Syntax: mv [opts] <source> <dest>

Parameters:

source Source file pathname.

dest Destination file pathname.

Options:

-h Built-in help
-?

-o Overwrite any existing destination file.

Notes:

MV actually moves a file's directory entry from one directory to another. If the source file is a directory, MV will also correct its ".." entry to point to the correct new parent directory.

The format of the source file name is [<dirpath1>/]<filename1>. If <dirpath1> is not specified, it defaults to the current working directory.

The format of the destination file name is [<dirpath2>/]<filename2>. If either part of the destination file name is not

specified, it defaults to the corresponding part of the source file name.

MV moves the directory entry for the source file from directory <dirpath1> to directory <dirpath2>, and then changes the file name field of the directory entry from <filename1> to <filename2>.

If the source path and the destination path are not on the same device, MV uses the COPY command to copy the entire file between drives, and then deletes the source file.

Unless the -o option is specified, MV will abort with an error if <filename2> already exists in <dirpath1>. If the -o option is specified, and <filename2> is a file, MV will delete the old file. The exception to this is when <filename2> is the name of an existing directory. In this case, "/<filename1>" will be appended to <dest> and MV will again attempt to split the result into <dirpath2> and <filename2>. This sequence will repeat until <filename2> does not refer to an existing directory.

Notice the difference between these 2 commands:

```
mv /d0/src/test /d0/temp
mv /d0/src/test /d0/temp/
```

The first command moves the file "test" from directory /d0/src to directory /d0, changing its name to "temp" -- the final

file name is /d0/test. The second command moves the file "test" from directory /d0/src to directory /d0/temp, without changing its name -- the resulting file is named /d0/temp/test.

If /d0/temp already exists, and is a directory, the first command will behave identically to the second. This occurs because MV will move a file into an existing directory rather than try to delete that directory.

If /d0/temp already exists, but is not a directory, the first command aborts with error 218 (file exists) and the second aborts with error 215 (bad pathname).

RECOVER

Function: Recover "lost" files on an OS9 file system.

Syntax: recover [opts] mapfile dirname

Parameters:

mapfile A file output by DCHECK for the device to be recovered.

dirname An existing directory on the device to be recovered.

Options:

-h Built-in help
-?

-q Querry mode. Display file recovery information but do not update file system.

Notes:

RECOVER locates lost file descriptor sectors and file data on an OS9 file system. It groups previously lost data into files, and creates directory entries for those files in a user-specified directory. The resulting new directory entries recover the lost files.

The program uses a work file produced by DCHECK as input, and must not be run unless DCHECK reports that the file system is intact (NOTE: a file system can be

intact even if it has lost files). The work file to be used has the name 'dcheckXX0', where XX is the hexadecimal process ID of the DCHECK process that created the work file.

It is very important that no programs modify the file system to be recovered between the time when DCHECK is executed and when RECOVER is executed. Users can ensure this by not running other processes concurrently with either DCHECK or RECOVER, and by running RECOVER immediately after running DCHECK.

RECOVER builds a work file indicating which sectors are lost. It then makes three passes over the lost sectors, attempting to recover files during each pass.

The first pass locates directories. If RECOVER finds a directory, it restores it and terminates, asking you to re-run DCHECK. If an entire sub-tree of directories is lost, RECOVER automatically restores the recognisable directory that is highest in the tree. RECOVER has to exit, and requests that you re-run DCHECK, because restoring a directory invalidates the information in DCHECK's work files.

The second pass locates files for which a file descriptor still exists. This pass will not be executed unless no directories were found in pass 1. RECOVER finds and restores as many files as possible during this pass. If any files have been found at

the end of the pass, RECOVER terminates and requests that you re-run DCHECK.

The third pass creates a file for each contiguous group of lost sectors. This pass will not be executed unless both pass 1 and pass 2 fail to recover any data.

All recovered files are appended to the specified directory. This directory must have existed at the time DCHECK was run, or RECOVER will attempt to recover it, too!

RECOVER automatically generates names for recovered files. These names have three forms:

- An OS9 module name. This will occur only if "lost" file descriptor sector is located for a file that begins with an OS9 module. The file is given the name of the initial OS9 module.

- A name of the form:

  XnnnnnnR

  Here, a "lost" file descriptor sector was found but the file did not begin with an OS9 module. 'nnnnnn' is the hexadecimal LSN of the file descriptor sector.

- A name of the form:

  ZnnnnnnR

  Here, a group of "lost" data sectors without a file descriptor sector have been assembled into a new file. 'nnnnnn' is the hexadecimal LSN of the first data sector in the file.

RECOVER can be interrupted safely at any time by pressing the [BREAK] key once. The program responds by displaying a message to acknowledge the BREAK, and continues processing until the file system is stable. Pressing [BREAK] a second time will terminate RECOVER immediately (see WARNING below).

Warnings:

WARNING: RECOVER NEEDS EXCLUSIVE ACCESS TO THE FILE SYSTEM BEING RECOVERED. DISK ACCESS BY OTHER TASKS (E.G. EDITORS, COMPILERS, BBS, ETC) CAN CONFUSE THE PROGRAM, RESULTING IN LOSS OF DATA.

WARNING: THE RECOVER ALGORITHM WORKS ONLY FOR CLUSTER SIZES OF ONE SECTOR. DO NOT USE RECOVER ON DISKS WITH OTHER CLUSTER SIZES.

WARNING: RECOVER CAN BE INTERRUPTED SAFELY BY PRESSING THE [BREAK] KEY EXACTLY ONCE. THE PROGRAM WILL TERMINATE IMMEDIATELY IF

[BREAK] IS PRESSED A SECOND TIME. SINCE THE FILE SYSTEM MAY NOT BE STABLE WHEN RECOVER EXITS IMMEDIATELY, DATA MAY BE LOST.

Examples:

To recover lost files from hard drive '/h0', once the file system is intact, using a DCHECK work file created by process 3 and stored on ram disk '/r0', use the command:

```
OS9: recover /r0/dcheck030 /h0
```

ZAP

Function: Lose a file or directory.

Syntax: zap [opts] [file ...]

Parameters:

file	The pathlist to a file to be lost.

Options:

-h	Built-in help
-?	
-z	Zero out file descriptor

Notes:

ZAP loses a file or directory by clearing its directory entry without deallocating its sectors. Sectors allocated to the file will show up as "in allocation map but not in file structure" during subsequent runs of the DCHECK utility.

This utility can be used to make emergency file system repairs when DCHECK reports a serious problem: use ZAP on any damaged files, then recopy those files from a backup or use RECOVER to restore them.

ZAP leaves all sectors in the specified files allocated, preventing OS9 from using the (possibly defective) region of the disk where the files were stored.

Normally, ZAP does not alter the data in either the file descriptor sector or the data sectors of the file. If the '-z' option is specified, ZAP will zero out the file descriptor sector. This is useful when the data in the file descriptor sector is corrupt and you don't want file recovery utilities to inadvertently re-use it.

WARNING: ZAP RENDERS THE DATA IN ALL SPECIFIED FILES INACCESSIBLE.

Examples:

When preparing to recover files from a hard disk, you should eliminate any files stored in defective areas of the disk. The DCHECK utility will identify these files, but they should not be deleted because doing so would allow RECOVER or other OS9 utilities to store data in the defective area. Instead, the files should be eliminated with ZAP:

```
OS9: zap -i /h0/cmds/badstuff
OS9: zap /h0/util/foo
(etc)
```


A.0 Technical Workshop

Understanding certain technical information about OS9 and the File Recovery System utilities will help make you a "power user". We encourage you to familiarize yourself with this appendix, and recommend several other sources of information about OS9:

Inside Level 2 OS9
by Kevin Darling
Frank Hogg Laboratory
770 James St.
Syracuse, NY 13203

OS9 Level Two Operating System
Tandy Corporation
(manual included with Level 2 OS9)

Start OS9
by Paul K. Ward
Kenneth-Leigh Enterprises
1840 Biltmore St. NW Suite 10
Washington, DC 20009

The Complete Rainbow Guide to OS9
by Dale Pucket and Peter Dibble
Falsoft, Inc.
The Falsoft Building
9529 U.S. Highway 42 P.O. Box 385
Prospect, KY 40059

Extensive OS9 databases can be accessed by modem on the Compuserve, Delphi, and Genii information services. You can also leave

mail for Burke & Burke or other OS9 users by sending it to the following information service accounts:

Delphi: COCOXT

Compuserve: 72240,304

At this writing, the information services charge about $10 per hour of connect time.

A.1 RBF File System

All computers use a set of rules to store data and programs on a disk. The set of rules varies between operating systems. For OS9, the set of rules is called the Random Block File system, or RBF.

The rules of RBF define a few simple objects: the ID sector, the bitmap, file descriptor sectors, and segments. RBF also defines a few rules for relationships between the objects. The objects on a particular disk and the relationships between them are sometimes called a File System.

All of the relationship rules are built into a part of OS9 called the RBF File Manager. The file manager is visible as "RBF" in the output of the MDIR command:

```
REL   BOOT  OS9P1     OS9P2 CC3GO
INIT  RBF   CC3DISK   .     .
```

Disk utilities like DIR, DEL, COPY, and

RENAME don't need to know very much about the relationship rules; they simply have the RBF module do all of the dirty work.

Some utilities (like REPACK) have built in knowledge of the RBF rules. Utilities benefit from this knowledge since it enables them to manipulate objects in ways that are not built into the RBF module. One disadvantage of this built-in knowledge is reduced "portability": the utility may only work on one type of computer.

A.1.1 The ID Sector

The first 256 bytes of data stored on any RBF disk describe the size of the disk, the size of the bitmap, where the first object is stored (the ROOT DIRECTORY), and several other related facts to OS9. This allows OS9 to automatically adjust to different disk sizes, etc.

A.1.2 The Bitmap

Each object stored on a disk takes up space. The bitmap tells OS9 which disk sectors are being used to store existing objects and which sectors are available to store new objects.

Typically, each bit in the bitmap represents one sector of information on the disk. The appropriate bit is set when the corresponding sector is in use.

A.1.3 File Descriptor Sectors

A disk file is made up of a file descriptor sector and zero or more segments. The file descriptor sector identifies all of a file's segments, and stores other information about the file including its attributes.

The attributes of a file determine how it can be used. The attributes are a single byte, with each bit enabling a particular use of the file. One bit, the "D" bit, identifies the file as a DIRECTORY.

A directory is just a file with the "D" bit set. This bit tells OS9 that the data stored in the file's segments describes the location of other files on the disk.

A.1.4 Segments

The actual contents of a file are stored in its SEGMENTS. A segment is a group of contiguous sectors; it may be located anywhere on the disk. There is a limit to the number of segments that a file can use, and also a limit to the size of each segment.

Since a segment can be anywhere on the disk, a file with several segments might be spread out all over the disk. The file descriptor sector links all of the segments together in a specific order, and OS9 moves from segment to segment automatically as

you access a file.

A directory is just a special type of file. The data in a directory's segments is divided into DIRECTORY ENTRIES, each of which gives the name of a file and the location of its file descriptor sector.

A.2 File Recovery Techniques

When files are lost, it is because one or more of the objects that make up the RBF file system are damaged.

The utilities on the File Recovery System diskette, when combined with the OS9 DCHECK utility, provide all of the tools that you will need to detect and repair most types of file system damage.

DCHECK is a powerful utility that identifies most types of file system damage through specific error messages.

The DCHECK utility generates four types of error messages, each of which indicates a different type of file system dammage. You can repair or neutralize each of these types of damage using the tools on the FRS diskette.

The DCHECK error messages are "bad FD segment", "previously allocated", "not in file structure", and "not in allocation map". This section explains what to do when you encounter each type of message.

A.2.1 Bad FD Segments

The first type of DCHECK error message indicates that a file's file descriptor sector is corrupt. This type of error can often cause DCHECK to gnerate other, false errors.

You can neutralize this type of damage with the '-z' option of the ZAP command. For example, if DCHECK reports a bad FD sector for the file /h0/cmds/badfile, you should respond with:

```
OS9:zap -z /h0/cmds/badfile
OS9:
```

The error will not recur the next time you run DCHECK.

A.2.2 Previous Allocations

This type of DCHECK error message indicates that the same sector is part of two or more files. Usually this means that one of the files is intact and the others are currupt.

You can determine which files are involved by looking up the sector numbers (specified in the error message) in the section of DCHECK's output titled "Pathlists for questionable clusters". You should then examine each of these files manually, and ZAP all of the corrupt ones. In some cases, none of the files will be intact and you will have to ZAP all of them.

For example, if DCHECK reports previously allocated clusters in files /h0/sys/font1 and /h0/usr/test/foo.c, and you determine that /h0/usr/test/foo.c is corrupt but /h0/sys/font1 is intact, you should respond with:

```
OS9:zap -z /h0/sys/font1
OS9:
```

The error will not recur the next time you run DCHECK, but you will probably see several new "not in file structure" errors; these can be ignored.

NOTE: WHEN SEVERAL FILES SHARE SECTORS, USUALLY ONLY ONE OF THE FILES IS INTACT. THE RECOVER PROGRAM WILL SALVAGE ONLY UNCORRUPTED SECTIONS OF FILES CORRUPTED IN THIS MANNER.

A.2.3 Not in File Structure

This type of error indicates a lost sector. You will generally not see this error message when running DCHECK as described in this manual, because of the '-b' option.

The RECOVER program automatically handles all errors of this type by recovering lost sectors. Remember, you shouldn't use RECOVER until no other error messages are displayed by DCHECK.

A.2.4 Not in Allocation Map

Usually this error indicates that data has not yet been lost, but is likely to be lost soon because some file is using a sector that OS9 believes is available for use by other files.

The error message includes the number of the problem sector. Very often this error will occur for blocks of sequential sectors. For example, a series of error messages might indicate that sectors $001234-$001250 are not in the /h0 allocation map. The BA utility shoud be used to force allocation of the specified sectors:

```
OS9:BA /h0 1234-1250
OS9:
```

The error will not recur the next time you run DCHECK.

A.3 Closing Remarks

We hope that you will enjoy these and other creative uses of the File Recovery System utilities. Thank you for your support of Burke & Burke.

