DiskSalv

Dave Haynie

COLLABORATORS

	TITLE : DiskSalv	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Dave Haynie	August 4, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

DiskSalv

1.1 DiskSalv Help

DiskSalv 3 Release 12.18 Commercial
Published (German) by
Stefan Ossowski's Schatztruhe
DiskSalv 3 Release 12.19 Commercial
Published (English) by
Intangible Assets Manufacturing
Welcome to the DiskSalv 3 electronic manual. This is a complete on-line AmigaGuide manual for DiskSalv. It can be accessed directly via AmigaGuide readers, or as called up by the DiskSalv program's help facility.

Table of Contents

1.2 dave

Dave Haynie has been involved in the Amiga community since the dawn of the
Amiga. He was an engineer on the Commodore C128 at the time Commodore bought Amiga. He started using the Amiga in 1985, as soon as he could get his hands on one. He bought one in 1986, and has been programming it ever since.

As an Systems/Hardware Engineer at Commodore, Dave was the chief engineer of the Amiga 2000, Amiga 2630, the Zorro III Bus Specification, Amiga

3000+
, Amiga 4091, and the
Nyx prototype
, and a leading member of the
engineering team on the Amiga 3000 and Amiga 4000. Dave had a number of

really cool projects
in the works when Commodore went under.

Independently of Commodore, Dave has been involved in a number of Amiga projects. As well as DiskSalv, Dave wrote SetCPU, the popular 68030 MMU tool, and several other example programs. Dave has written extensively, for magazines including AmigaWorld, Amazing Computing, Amiga Sentry, .Info, AmigaWorld Technical Journal, Compute!, Amiga Shopper (UK), and most recently, Amiga Format.

In less technical times, Dave resides in the great state of South Jersey, with his wife Liz, kids Sean (3) and Kira (1), dog Auryn (Borzi), and cat Iggy (Black & White, footwarmer). When he’s not on the computer, Dave has too many other interests. He practices Aikido regularly, he’s into photography, video (see his first film,
the Deathbed Vigil
), cycling,
canoing, swimming, woodworking, Japanese knives, writing (technical and songs), modern Rock music, and good beer. Since the demise of Commodore ("well, at least I can have a life now"), Dave’s been working at Scala, Inc., and learning to play keyboards.

1.3 a3000plus

In 1990, Commodore was nearing completion of the first major upgrade to the
Amiga chip set. Code named "Pandora", and later dubbed "AA" (because of the

AAA
project, already underway), this chip set would boot the basic graphics ↩
capabilities of the Amiga considerably, while retaining full register-level compatibility with the ECS and original Amiga chip sets.

Ultimately, Commodore needed a test system for these new chips, and so they naturally assigned Dave Haynie to the project. Not satisfied to just build an "AA3000", Dave looking into building an all-around better system which included the new chipset.

In February of 1991, the first A3000+ booted up Workbench. The new chips ran the existing AmigaOS almost without incident. In addition to the AA chips, this first 3000+ had a digital signal processor, the AT&T DSP3210, built in as a local bus coprocessor. The DSP3210 and the Amiga were a match made in heaven. The 3210 was a local bus master, allowing for DSP systems to be built without the expensive and limiting SRAM of earlier designs. The 3210, at 50MHz, crunched 32-bit floating point at up to 25MFLOPS, five times faster than the 68040. AT&T has a full fledged, multitasking, multiprocessing DSP operating system for the 3210, which used an arbitrary general purpose OS as a host. The Amiga’s low overhead, near realtime OS was a perfect mate for AT&T’s VCOS.

Of course, prototypes will be prototyes, and the DSP never worked on the Rev 0 edition of the A3000+. But everything else did. For Rev 1, a very extensive DSP audio system was put into place, including hardware CODECs for 16-bit stereo I/O at up to 48kHz and phase-correcting telecommunications, for V32 modems.

After Rev 1, Commodore Engineering management was changed by the then president of Commmodore International,
Mehdi Ali
. The new VP of
engineering, Bill Sydnes, was opposed to the A3000+, and virtually every other project underway at the time -- no use making the previous administration (the folks who brought you the A500, A2000, and A3000) look good. The final revision of the Amiga 3000+ was a scaled down version, as mandated by the administration. A flaw in some custom DSP support logic, built into the new A3000-architecture DMAC chip, made the DSP a problem.

The DSP lived on for awhile, despite management. Dave Haynie worked on his own time to get systems reworked, and work out any additional bugs, in the DSP hardware. Jeff Porter, onetime Director of New Product Development and the other driving force behind the DSP, managed to keep the software development funded. Eric Lavitsky, DSP expert, consultant, and longtime Amiga supporter, did the actual VCOS port. This port was, in fact finished.

And the DSP had a kind of afterlife. After Bill Sydnes was fired, Lew Eggebrecht took over Commodore Engineering. While not an amazing leader, Lew did turn a number of projects on that were floundering as skunkworks efforts necessarily hidden from Sydnes. Dave had proposed a DSP board be made for Zorro III, and Lew put two engineers on it full time. Although Commodore never built the resulting board, the design was nearly complete, and it was build by a company that licensed the design before Commodore went under.

1.4 nyx

The most advanced project ever attempted at Commodore was the creation of the Advanced Amiga Architecture, or AAA. Started in the late 80's, AAA was an effort to build a new Amiga architecture that was once again head and shoulders above the mainstream. AAA was a major advance. It would deliver 32 and 64-bit systems, using DRAM or VRAM. Many new graphics modes were supported, including 24-bit, HAM10, and compresses 8 and 24-bit modes. The blitter and copper were fully 32-bit, and the copper could feed the blitter. Graphic resolution went up to 1280x1024 noninterlaced, and the pixel clock could change on a line-by-line basis, to support hardware promotion of older screen modes. Audio was extended to 8 channels, with sampling rates up to 100kHz at 16bits/sample. The floppy disk interface was fast enough for 4MB floppies, 150KB/s CD-ROMs, Digital Radio, and other serial streams, and decoding could be done on-chip or in software.

In 1992 Dave Haynie designed the "Nyx" prototype, which was the first home for the AAA chips. Based on the A3000 architecture and lots of programmable logic, three working systems were built. When Commodore stopped funding the AAA project, critical chip revisions had been released to tape, but not yet made. The existing AAA chips delivered 24 bit graphics at high resolution, demos set up the copper feeding the blitter, doing CPU-less animations. The next rev was supposed be enough to boot the AmigaOS.

1.5 deathbed.vigil

The Deathbed Vigil and other tales of digital angst
by Dave Haynie

Set the way-back machine for April, 1994. Everyone was worried about the continued existence of Commodore. I had been away, interviewing for new jobs in Texas, so I came in, first time that week, on Tuesday, April 26, 1994. Rumors were running rampant about a bigtime layoff happening the next day. We in Engineering had already been on a major league skeleton crew since the summer of 1993, so this was clearly a sign of the beginning of the end.

So, when I woke up Wednesday, not knowing with any certainty if I'd have a job to go to tomorrow, I thought about videotaping Commodore. After all, this whole Amiga thing, which ran far beyond Commodore, the Amiga community, nearly all my active interpersonal relationships, and in some sense, the last vestige of the real small computer industry; once full of excitement and new ideas, but by this time more concerned with perpetuating and recreating obsolete, "best of the 70s, as long as UNIX isn't considered" computing. It was one of those ideas you think about, say "hey, wouldn't thi be cool", but then dismiss as soon as there's an obsticle.

So I set out to do some taping. Fortunately, I had three batteries charged for my Sony TR-7 8mm camcorder; recharged after my trip to Texas. And fortunately, K-Mart had blank 8mm tapes for sale. So I went to Commodore, and proceeded to do a walk-around of the Commodore building.

Pretty early on, it was clear that the layoff was happening. All but about 30 people were layed off; I was one of the "still employed", it was less

clear who the lucky ones were. At lunchtime, we went to our Mexican place, Margarita's, for the last big layoff party. There, many things were said about the Commodore management, some of it on-camera.

At the layoff party, Randell Jesup told me of a "Deathbed Vigil" party that he and Bryce Nesbitt were throwing, on Saturday. When Commodore bought Amiga, the good folks at the original Amiga company in Los Gatos, CA, held an "Amiga Wake" party. This proved premature, if in retrospect technically correct; Randell didn't want to make the same mistake. So I filmed the party, where all kinds of cool things took place: interviews, tales of the golden and not-so-golden years of Commmodore-Amiga, a burning of the L.B.M. effigy (some associate this with ex-President of Commodore, Mehdi Ali), smashing of keyboards, the "Chicken Lips Blues" song (performed by Mike Rivers, written by attendees), and other great events. Some strange happenings, post-party, were also filmed at Commodore.

Once done, I had to figure out what to do with this 4-5 hours of video. I decided to make a real, for sale videotape, and to try to tell a bit of the story of What Went Wrong, along with the antics, anger, info, and catharsis of this time. I realized I wanted to have on tape some small piece of this amazing thing I had been involved with for 10 years, and I figured fans of the Amiga might want a look too. So I set out to really make my first film.

And along the way, I decided I wanted to know: is Desktop Video real. I never did any video stuff at Commodore. So I wanted to know, could a novice videomake sit down with a consumer camcorder and deck, an Amiga, some plug ins and the right software, and actually make a good video. So I put together my system, including:

Amiga 3000+
prototype
Scala MM300 authoring system (provided by Scala)
Scala EE100 LANC/IR controller (provided by Scala)
SuperGen 2000 (borrowed)
GVP TBCplus (borrowed)
JVC HR6900 SVHS deck (paid in cash)
Deluxe Paint IV

Over the course of four months, I put together the video. I added various bits of information gleaned from conversations with past and present Commodore employees, and other folks "in the know". Mike Rivers provided me with some original music, and I wrote lyrics to one of his songs, which I affectionately entitled "F.Y.M.". We drank beer, mixed it, and I became a Rock Icon. NOT. Anyway, if you like the Amiga, and want some idea of what went wrong, you can still order "The Deathbed Vigil and other tales of digital angst" from
Intangible Assets Manufacturing
.

1.6 cool.projects

There seems to be a general feeling in the Amiga community that ↩
Commodore's
engineering teams spent years developing Really Amazing Things, only to

have Marketing deep-six them on the verge of production. This does occasionally happen; the A3000+ is a good example of this. What was more often the case, though, were projects done on the side, "skunkworks" projects, if you will, that were often cool, but didn't get very far, due to varying management support, available funding, and the amount of "copious spare time" available to the project's owner.

A few of Dave Haynie's less successful projects, over the years, included:

A2630 This one actually made it out the door. For six months, it was a funded skunkworks project. Nearly overnight, it became A Real Product.

BIGRAM A 16MB board for the A2630. Two were built. Hey, in 1988, 16MB was lots of memory.

FASTRAM An 8MB board with fast page support for the A2630. As it turned out, this was a bit too complex to do in PAL logic at the required speeds, but it was a good design exercise.

BIGRAMZ3 This was done in about two weeks, start to finish, as a Zorro III design example for the 1991 DevCons. This is a 64MB Zorro III memory card that supports Zorro III burst. It benchmarkst at about 80% the speed of local bus memory, a fairly impressive accomplishment given the less-than-ideal Zorro III interface of the A3000 architecture. About four of these exist.

A2631 After the A3000 went out, Commodore was still, strangely enough, shipping lots of A2500/30s. Certain niches wanted the larger box of the A2000. Every A2500 got an A2630 and A2091 board. One Friday, over beer and Mexican food, Dave Haynie and Greg Berlin got the idea that this was stupid, in the light of the A3000 architecture. So the next week, Dave cranked out a replacement, based on the A3000 architecture, which we called the A2631. This was an A2000 CPU socket board with Buster, RAMSEY, the DMAC and SCSI chip, 68030, and 68882. It cost less than the A2630, delivered high performance SCSI, and could take 16MB of RAM. Management wasn't interested, even though it would have saved money. Two prototype boards were built.

Gemini This was a multiprocessing board, designed to test and stress the features of the Level II Buster chip (Rev 8 and beyond). The problem with inventing your own expansion bus is that you have to build everything for it. So for fun, this board did something interesting; it had two 68030s, each with 4MB of RAM and independent Zorro III access. Two of these were built, but the project resources were pulled before it was debugged. Had it been

developed fully, this could easily have helped to debug the Buster chip before the A4000 shipped.

Acutiator Only a paper design, Acutiator was an effort to specify a whole new system-level architecture, replacing the A3000 architecture used in all A3000 and A4000 systems. The goal was a cost efficient, high performance architecture that could deliver anything from midrange systems (about midway between A4000 and A1200) on up to fully professional Amiga system heretofor nonexistent. Haynie originally designed a new "Amiga Modular Interconnect" bus, but adopted the fairly similar PCI bus once it was announced. The main idea was to make "highly modular" Amiga systems, wherein the system board design was independent of CPU or graphics subsystems. A small amount of design work had been done on this, but it was largely ignored by management.

SCARAB The SCARAB board, the last thing Haynie worked on at Commodore, was an effort to build a high performance graphics card based on off-the-shelf SVGA chips. The card ran a PCI bus locally, with bridged to Zorro III and to the video slot. With the video slot interface, Amiga chip graphics could be converted, in realtime, to PCI cycles, which wrote the SVGA graphic memory, in a window controlled by SCARAB registers. In essence, this was a programable "flickerFixer" that could handle any scan rate. The board could also support "hybrid" graphics modes, where in the Amiga chips were still used, but went into a very slow scan mode, so they could put out 1024x768 at 8 bits in slowscan, which would be converted to 72Hz noninterlaced by SCARAB (this is somewhat like "Hedley hires", an easy addition to the AmigaOS). RTG drivers would ultimately hit the board, directly, over Zorro III. Lots of design work went into this, but it became pretty clear there was no money left to actually build any of it.

1.7 schatztruhe

DiskSalv 3 is published in the German language by Stefan Ossowski's Schatztruhe:

Stefan Ossowski's Schatztruhe
Gellschaft fur Software mbH
Veronikastr. 33
45131 Essen
Germany

Telefon 02 01/78 87 78

Telefax 02 01/79 84 47

1.8 iam

DiskSalv 3 is published in the English language by Intangible Assets Manufacturing:

Intangible Assets Manufacturing
828 Ormond Avenue
Drexel Hill, PA 19026-2001
USA

For information on IAM products, etc.

email: info@iam.com
web: http:://www.iam.com

1.9 whatsnew

The crash-on-exit bug in Version 12.16/12.17 has been eliminated.

The fix-in-place routines have been enhanced. The result of this is that directories with problems are much more likely to be fixed, rather than eliminated. DiskSalv's ability to deal with various kinds of bad blocks linked into the active disk partition is also much improved.

New startup and stack extension code solves stack overrun problems completely. This may have some small effect on performance, but it should eliminate at least half of the problems folks have been having with DiskSalv.

A new command option,
SKIPDEVS
, has been added. This is used to keep DiskSalv from examining specific DOS devices.

What was new:

Version 12.16/12.17

1.10 version1617

Big improvements to the DiskSalv.guide file.

Full support of AmigaOS 2.1. Previously, bugs in the V34 version of the AmigaGuide.library, both in DiskSalv's calling conventions and the aspects of DiskSalv.guide format, caused the on-line manual to be unusable in most 2.1 installations. A workaround for the V34 function

bugs, coupled with reformatting of this guide file, makes 2.1
conventions work.

Added additional device checks have been added in the device listing
routines. This is designed to eliminate Enforcer hits when DS3 is run
on systems with certain devices, like NFS volumes, that were confusing
it before.

Many recursive routines rewritten to use much less stack, and in some
cases, less memory all around.

The output window wasn't being built properly after a "quick" scan; it
was basing itself on the existing size of the window. Now it uses the
predefined maximum window size.

Improved the block reference checks in the fix-in-place routine. This
improves the ability of DiskSalv to flag bogus disk block number
references.

Fixed a bug in the Best-Guess analysis that could cause no guess to be
made. This could result in a volume being labelled "BEST", and no
proper analysis being run. This last effect was a side effect of a
missing test on the setup window, which determines when a disk scan
run is safe.

Improvements to the input window better support NON-DOS devices, and
now properly prevent DiskSalv from running until a proper file system
is set by the user, in the event a possibly-valid alternate file
system is found on the disk (MS-DOS, alternate AmigaDOS file systems,
etc.).

1.11 dsforward

The DiskSalv Story

A little over nine years ago, I got my first AmigaDOS disk error. I was
writing some program or another for my brand new Amiga 1000, when the disk
failed. It was, of course, my only copy of this masterwork, and I needed it
back at all costs.

Much has been said at the time of just how robust the Amiga
file system
was. So the next day, I went over to the Software department at ↩
Commodore
to get their disk repair tool and get back to work on my great new thing.
Much to my dismay, there was no disk repair program. I left with a copy of
something called DiskEd which might help.

Some time later, the original intent nearly forgotten, I released a program
called DiskSalv V0.9, which soon became popular in the small but growing
Amiga community. It could handle any
device
, as long as it was floppy
unit 0, 1, 2, or 3. Even Commodore's CATS group used it, as
DiskDoctor

was still in the works, and couldn't legally be given out, even in prototype form. Over the years, DiskSalv 1 grew to support arbitrary AmigaDOS devices and much more sophisticated recovery methods.

Moving to DiskSalv 2

At the end of 1989, I started working on a full upgrade of DiskSalv. This program would use Intuition to ease user interaction. As time went on, the program grew, and so did the complications, as more and more things I attempted could not be done easily or cleanly in the AmigaOS 1.3 system.

By December of 1991, the 2.04 operating system was being finalized, and I had committed DiskSalv to 2.x-only operation. In June of 1993, I released DiskSalv 2, which provided much more sophisticated scanning routines, fix-in-place modes, and an AmigaOS 2.x compliant Intuition driven user interface, based on the evolving Amiga style guidelines. It was one of the first programs to support both Localization and AmigaGuide, though I never finished a guide for it. It also supported all AmigaDOS file systems, six different types at that time. Several releases later, DiskSalv 2 is quite stable, and continues to receive small enhancements.

And Finally, DiskSalv 3

Which brings me to DiskSalv 3. When I released DiskSalv 2, it was already rather apparent that a sophisticated GUI-driven program required better documentation that I could provide in a simple ReadMe file. I had originally intended, and offered, to provide a printed manual for DiskSalv 2. As I went through the feedback from the first DiskSalv 2 releases, I developed many of the enhancement ideas I had been holding back on to get DiskSalv 2 out the door. I decided then that rather than just offer a semi-commercial manual for DiskSalv 2, I would offer a complete program upgrade for the same price.

The first release of that is what you have before you. It incorporates a number of ideas in GUI and disk repair I couldn't or didn't offer in DiskSalv 2. I will continue to update DiskSalv 2, but my emphasis from now on will be on improving DiskSalv 3. Many new features are already in the works. Some are even fairly well developed, but left out of this initial release, which is already at least three months later than I had hoped.

DiskSalv 1 and, for the most part, DiskSalv 2, were programs you hoped never to need. DiskSalv 3 is one you may use more often, as it adds prevention to the cure of the previous releases.

-Dave Haynie

April 20, 1995

1.12 intro

DiskSalv is a disk recovery program. Its main purpose is to ↩
recover

AmigaDOS disk integrity when a disk fails, or when impossible, the data from a failed disk. The name DiskSalv is short for Disk Salvage. Originally, DiskSalv's only function was to extract as much data from a failed disk as possible and copy this information to another disk.

DiskSalv 3 has extended this function in various ways. It can recover deleted files from an undamaged disk, which is often a more common need than failure recovery. In many cases, DiskSalv can fix a damaged disk in-place, rather than copy out its contents to another
volume
. In
these days of multi-gigabyte hard disk drives, that's an important concern.

Finally, DiskSalv 3 adds a number of related features. It can find

partitions
on a disk, even when AmigaDOS can't. It can report errors on a disk without repairing them for you. It can backup an AmigaDOS
volume
to any AmigaDOS disk or tape device.

Topics:

1.13 goesbad

The AmigaDOS
file systems
are not very tolerant of even small defects in a disk's format, and will reject damaged disks. There are a number of things that can cause problems with a disk. Physical damage is perhaps the most obvious, and the worst kind. Fortunately, this is common only on floppy disks. Mechanical abuse, magnetic contamination, or just prolonged wear can cause physical errors on a floppy disk. Hard disks can fail in the same way, though they last much longer and aren't normally subject to drops, coffee spills, or other physical damage. Physical errors are called

hard errors
.

Much more common, especially on hard disks, are
soft errors

. A
soft error
is any disruption in the file structure of a disk that is not due ↩
to
physical damage. DiskSalv can very effectively deal with such errors.

1.14 notfix

On the Amiga, a program called the
file system
is responsible for
communication between the Amiga's DOS Library (eg, the Amiga's Disk Operating System core) and the
device driver
program, which serves
to abstract the function of any specific disk management hardware. Disks appear in many forms. Some are addressed via high-level (e.g., logically mapped) protocols such as the SCSI (Small Computer System Interface) or the IDE (Integrated Drive Electronics) protocol. Others may be supported at a lower level, addressed in terms of block, sector, and offset. Still others may exist only as an area of memory. The device driver allows any disk to be addressed as a linear array of disk blocks, and the file system only needs to work in these terms.

The file system is reasonably good at detecting errors. It maintains a bitmap of disk blocks it has used, and a flag that indicates if this bitmap is valid or not. Before making any changes to the disk structure, the file system marks the bitmap as invalid. It then make the requested changes, and finally, marks the bitmap valid again. If an error of any type, be it hard or soft, occurs during a modification, the bitmap will still be marked as invalid the next time the file system starts up.

The file system uses this condition to launch a routine called the disk validator. If AmigaDOS is running this on one your
partitions
, the Info
command will report Validating for that
partition
. During the validation
process, the file system checks out every object that can be reached by walking the disk structure from the disk's root block. As long as no bad blocks are found here, a new bitmap can be constructed. If a bad block is found, it must be considered unsafe to allow any more writes to that disk until the problem is resolved.

In most cases, the file system can do nothing to fix the problem, so it simply sets the volume to read-only, in cases of small damage, or NDOS (not understood by AmigaDOS) if the volume doesn't make any sense. It is considered beyond the scope of the file system to do much more than this. It would certainly be possible for the file system to be as clever as DiskSalv or other disk repair utilities, but there's no good reason for this. There is no advantage in locating sophisticated repair functions in the file system. The disadvantage, among others, is size: the file system is roughly 25K in size, while DiskSalv 2 is currently almost 120K, while DiskSalv 3 is approaching 200K. Even without a full GUI and with assembly

language downcoding, this would be a memory burden.

Most operating systems include a disk repair tool of some kind, though these tend to be rather simplistic. A program called
DiskDoctor
was once
included with the Amiga Operating System, but it had a number of problems, and was generally less than reliable.

1.15 canhelp

DiskSalv is designed to analyze a troubled disk and attempt a fix. ↩
For
every operation, it builds a consistent model of the disk in memory, and then uses that model to salvage the disk. DiskSalv can do something for any kind of error, and in many cases it can completely restore the disk to working order. The success depends on what happened to the disk, of course. When a disk or file can't be repaired, DiskSalv can still attempt to restore the data to another
volume
.

Topics:

The Fix-in-Place Concept

Where are my bad files?

The Recover-by-Copy Concept

1.16 intro.fixinplace

The most desirable result of a DiskSalv run is the complete ↩
restoration of
a troubled disk. Ideally, the trouble is not severe and can be fixed without the need for anything to be eliminated from the disk. While this is certainly possible, this isn't always the case. There are times when DiskSalv must eliminate a file or, rarely, an entire directory, in order to correct a disk's structure. And on occasion, a disk may not be restorable at all.
In most cases
, though, DiskSalv can restore a disk.

Hard errors present the most trouble, since there is little a piece of software can do to correct a hardware problem. It may be able to work around it, though, depending on the location of the error. A less critical hard error will occur within a file. DiskSalv can restore the disk to read/write status by removing the file from the active disk structure. A hard error in a directory or other disk structure management block may be impossible to get around, since the block would have to itself be modified to fix the disk's structure.

Even if DiskSalv does fix a disk containing a hard error, the error will eventually come back, since it's a physical flaw in the disk. At present, DiskSalv won't be able to do much about this problem. The next release of DiskSalv 3 contains a disk block mapping capability which can map out bad blocks on most kinds of disks.

Soft errors
can be perfectly fixed as long as there's enough of the disk structure left after the crash to make the disk still viable. At the worst, a file may have to be eliminated to fix a
soft error
.

DiskSalv does not attempt to repair the structure of a file itself in-place, but it will allow such faulty files to be recovered as intact as possible to another disk
volume
.

A directory can be completely reconstructed from DiskSalv's internal disk model. DiskSalv will only need to eliminate a directory if it contains a

hard error
.

1.17 intro.wherebadfiles

As mentioned, DiskSalv's fix-in-place routines are designed to repair a disk's logical partition, such that the partition's file system can be safely run on the partition. Occasionally, there will be file eliminated in order to make this possible -- when DiskSalv eliminates the cause of a file system validator hangup, the validator should be able to do its job.

When files are eliminated, DiskSalv would like to offer them to you, in the DiskSalv file browser, giving you the option to salvage them to another volume. On occasion, though, you may not see the files you're expecting to see. This is due to the nature of the Amiga file system and the kinds of errors that take place. If the error is in a specific file, this file can be explicitly eliminated, and shown in the list. However, if the parent directory of file damaged, it may not list that file in its contents. In this case, Repair will not attempt to recover that file, it will appear to Repair as a deleted file.

After a Repair run, such files can still be obtained. Using the Salvage or Undelete functions, these files can be found and restored to another volume. If there are an excessive number of them, run the Unformat function. This performs a global Undelete on everything on the disk, though it can bring back lots of things that are not needed as well as the missing things that are important.

1.18 intro.recoverbycopy

When DiskSalv can't fix a file, directory, or entire disk in-place, it offers the user an option called "recover-by-copy". DiskSalv never actually erases a file unless instructed to, via Cleanup mode. When a fix-in-place operation can't fix an object, that object is simply eliminated from its parent directory, it is never actually removed from disk. Some DiskSalv modes also intentionally do not attempt any modification of the input disk.

In a recover-by-copy operation, the user selects an output device that's different than the input device. This can be another hard disk, a series of floppy disks, a RAM disk, a disk-based file, or magnetic tape, or any other AmigaDOS device. In essence, a recover-by-copy operation works like a copy or back-up function, except that it uses a specialized set of routines designed to extract data from a potentially faulty disk. Anyone familiar with DiskSalv 1 will recognize this as the only recovery mode supported in that program.

Whether a recover-by-copy mode was explicitly selected by mode, or forced as the result of something being eliminated during a fix-in-place run, the Output routines in DiskSalv allow the user to select the files that will actually be restored, as well as the output device they will be written to. The output can be written according to the AmigaDOS file structure, as long as the output device support it, or in the DiskSalv archival format, which can be written to any AmigaDOS device.

1.19 install

DiskSalv 3 is easily installed anywhere on a hard disk or floppy disk. It doesn't require any support files, though it does support several, notably this DiskSalv.guide file for on-line hypertext help linked to DiskSalv.

Topics:

From the Workbench

From the Command Line Shell

Common Installation Problems

1.20 install.wb

The DiskSalv 3 distribution disk is set up for easy Workbench installation.
Simply click on the Install icon to install everything in its proper place on a hard disk. This uses the standard Amiga installer program for the installation, which should be familiar to most users by now.

It's also a good idea to put the DiskSalv 3 executable and support files on a bootable floppy disk. This should be labeled DiskSalv 3 Boot, or perhaps Don't Panic, put in a safe place. Should a disk error of some kind ever damage your hard disk to the point of it not being bootable, such a disk can be a lifesaver. The MakeBoot script builds just such a disk. It copies necessary files from both the DiskSalv 3 distribution disk and your system disk to make this bootable floppy.

Please make certain that your boot time OS is the one installed. If your system come up in another OS, you make need a second disk to install the current version before the DiskSalv boot disk will function. For example, I have 2.04 ROMs on my Amiga 3000, but it boots AmigaDOS 3.1 on startup via Nic Wilson's Set040 program. I would need to create a 2.04 version of the DiskSalv boot disk or I would need a disk to boot my system into 3.1 before using a 3.1-based DiskSalv boot disk. Amiga 3000 owners who use SuperKickStart already have such disks for their systems in most cases.

1.21 install.shell

Expert users may prefer to install DiskSalv manually from their favorite command line shell program. This will just take a couple of seconds. In most cases, you simply need to pick a target directory and copy DiskSalv, DiskSalv.info, and DiskSalv.guide from the distribution disk. It's not necessary to copy the .info or support files, but both are useful in most installations. The DiskSalv.guide file can be located in the same directory as DiskSalv or along the AmigaGuide HELP path, if one exists.

1.22 install.problems

DiskSalv installation should be very straightforward. While the program can use some of the optional features of AmigaOS 2.1 or 3.0, it only requires AmigaOS 2.04. It will run on 512K systems, but it will require more memory to process large disks. Similarly, it will run with as little as 4096 bytes of program stack (the Amiga OS default), but more is suggested for processing large disks. The DiskSalv 3 icon sets a default of 20K for the stack, which should be enough for all but the largest disks.

If DiskSalv refuses to start up probably, a couple of things should be checked. Make sure the Workbench screen, or public screen you have assigned to DiskSalv, is at least 640x200 in size. Also make certain that the topaz 8 font is available (it's usually in ROM). DiskSalv attempts to adjust to the system's default font, but will drop back to topaz 8 if necessary.

1.23 quickstart

If you are interested in learning all everything written about DiskSalv, I recommend reading the rest of this manual. If, rather, you're interested in fixing a problem disk as soon as possible, the rest of this section may help you get going without an extensive knowledge of DiskSalv. Help is available by pressing the HELP key over any gadget or menu item.

Topics:

The Setup Window

The Scan Window

The Output Window

1.24 quick.setup

The DiskSalv program is started simply from the Workbench or command line. An introduction screen will be displayed, presenting the program version, release, and copyright information. Click on the Begin gadget or double-click the right mouse button to get the Input window, which is where DiskSalv always starts.

The Input window is where DiskSalv is set to a particular mode and given a disk to work with. In most cases, it's simplest just to drag the icon from your problem disk into the Input window, though the device can be selected via a requester instead, by clicking on the Device list gadget. DiskSalv will automatically select the best file system for processing. If you're absolutely certain DiskSalv has selected this incorrectly, the file system can be changed via the file system requester, by clicking on the File System list gadget. An aborted format attempt may cause file system types to be incorrect, but they rarely are otherwise. Selecting the wrong type can cause damage to the input disk's data.

The next step is to select the

Major Mode
. The Salvage mode runs a basic

recover-by-copy
operation. It will allow any file on the input
disk to be copied to an output disk as completely as possible. The
Undelete mode runs a
recover-by-copy
operation, but will only list
files that are deleted. The Repair mode is the basic
fix-in-place
function. Use this to fix a disk with any kind of error reported ↩
on it.
Finally, the Unformat mode is used to reverse the effect of a Format Quick
or an aborted full Format, as completely as possible.

1.25 quick.scan

Once the
device
and mode have been selected, the disk must be
scanned. This is the process by which DiskSalv builds a model of the input
device, activated by pressing the Scan gadget on the
input window
.
This causes a switch to the
scan
window. This window presents an
ongoing display of the scanner's findings, including a count of objects
encountered, a status gauge, and an item-by-item display of each
significant object encountered. Several different scanning phases may be
invoked as a part of each mode; the
fix-in-place
operation can
comprise as many as seven passes. A scanner run can take a minute or so on
a small disk drive, many minutes or even longer on a large drive. The
scanner display can be paused at any time, or stopped completely. It may
take some time to actually perform a stop operation, since DiskSalv will
not allow the scan to stop with a
fix-in-place
operation partially
complete.

1.26 quick.output

As long as a
fix-in-place
mode is selected and no identifiable
DOS objects must be removed from disk in order to fix it properly, the
scanner will terminate and offer the user a choice of going back to the

input window

for a new device or quitting DiskSalv altogether. If a

fix-in-place
mode must delete something recognizable as a file or directory, or a
recover-by-copy
mode is selected, DiskSalv will open the Output window once a scan has completed.

The primary feature of the output window is a pair of file list requesters. The requester on the left lists the directory structure obtained during the scan, while the requester on the right lists the files belonging to the current directory. When the output window comes up, nothing has been selected and the root directory is current, indicated by normal listview highlighting. Clicking on a directory name makes it current and displays any files it may contain, while clicking on a file selects that file for restoration. There are various options available for manipulating these objects; the most important is the set gadget, which selects the current object and, in the case of the directory requester, all of its children.

Once at least one file or directory has been selected, the output device is specified. This can be typed into the Output string requester, or selected via the file requester brought up by clicking on the Output list gadget. DiskSalv can rebuild part or all of a disk's structure on any file-oriented

AmigaDOS device
.

1.27 problems

While there are many problems that can disturb a disk structure, ↩
most of them seem to be pretty rare, fortunately. Of the errors that do occur, a couple of common types stand out. Most of these problems can be solved by the casual DiskSalv user, though a few require a bit more knowledge to properly address.

Topics:

Checksum Error

Key Already Set

Not a DOS Disk

Can't Find Volume

Can't Find Device

1.28 prob.error

One common disk error reported by the
file system
is a Checksum error.
Directory, file, and other disk management blocks contain a 32-bit checksum field. This is a number selected to cause the sum of all longwords in a block to result zero. If for some reason, the file system detects a block that doesn't sum to zero, it will make that disk read-only and notify the user.

It's very unlikely that the file system would incorrectly calculate such a checksum. Still, a checksum could be incorrect on disk for most any hard error, or any situation that causes a disk block to be somehow miswritten. DiskSalv's Repair mode will nearly always solve this kind of problem. Fixing this can result in a file being removed, but will never not result in a directory elimination unless there's a
hard error
detected.

1.29 prob.key

Another common error, Key Already Set indicates that the
file system
has somehow allocated the same block, or key in
Tripos
parlance, for
two different objects. This usually occurs when a directory or file header block is incorrectly updated, so it references the wrong block as a child or contents block. This can also happen if a disk block goes slightly bad due to a hardware problem, though this is usually caught as a checksum error instead. DiskSalv's Repair mode can fix this condition by eliminating one of the two objects referencing the block. It is generally possible to tell which object is the proper owner of any particular block.

1.30 prob.ndos

A more severe crash can result in a "Not a DOS Disk" message, ↩
which
indicates that the Amiga
file system
can not recognize a
partition
as a
valid AmigaDOS partition. This condition may simply be a matter of luck more than anything else. A crash that affects a file or a subdirectory block may not be fixed by AmigaDOS, but the disk will usually be recognized by AmigaDOS, though it will be created as a read-only disk (the file system wisely prevents modifications to a troubled disk). The same disk crash on a disk's
root bock
or initial block will confuse AmigaDOS. Without
the ability to process a disk, AmigaDOS marks it as NDOS. Of course,

hard errors
 or other other physical problems can cause this
too, and generally can't be helped by DiskSalv.

DiskSalv can usually solve this kind of problem, too, using Repair mode. If
the problem is with the disk's
root block
, DiskSalv can reconstruct that
root based on the results of a full scan of the disk. If the problem is
some damage to the disk's reserved area, DiskSalv may not be able to
quickly determine the disk's DOS type. The AmigaDOS file system stores a
disk's DOS type (Original File System, Fast File System, etc.) as a code in
the first reserved block. DiskSalv can use the
Best-Guess
 option in the
file system requester rather than a specific file system to determine the
DOS type automatically.

1.31 prob.novol

This is much like the "Not a DOS Disk" problem. In AmigaDOS, every

partition
 can be referred by either
volume
 or
device
name. A device name, such as DF0:, DH0:, etc. is usually set up at ↩
    boot
time, before AmigaDOS has initialized the disk. AmigaDOS device names
depend on the configuration of the Amiga system you're using. The volume
name is read from the disk, and depends on the disk itself. The floppy
drive is generally called DF0:, but it may contain volumes such as
Workbench: or DiskSalv3: from time to time. When a volume name can't be
found, that's an indication that there may be a significant problem with
the disk, the same kind of problem that causes "Not a DOS Disk" problem.
The Repair mode is suggested for this too.

1.32 prob.nodev

In the most severe failures, the
DOS device
 designator for a volume
is not present. This can only happen for automounted hard disk and similar
devices that follow the Amiga Rigid Disk Block standard (
RDB
). RDB
is a convention followed by nearly every Amiga hard disk device that stores
disk partitioning information in a standard way. This allows partitions to
be automounted at boot time, and makes it very easy for disks to travel
between different systems, even if they use different makes of hard disk

controller.

The down side to this, of course, is that the disk partitioning information is contained on the disk. Any force that can cause a disk crash on a a file or directory block can damage the RDB area, though since it's a small area, this isn't all that likely. Still, if it does happen, you're in more trouble than usual, since without the RDB, there is no description of a partition for DiskSalv to work from.

DiskSalv can, however, help you out here too. This is a more complicated process, and it's not a bad idea to read about the
Device Analysis
function and the
Device Editor
at this point for more information
on the device editor. Simply put, you want to run the device analysis function, which is called up from the
input window
by clicking
on the analysis button.

The Analysis window uses the Device Editor to let you specify the disk to search. This requires entry of the Exec-level device and unit to search, as well as some other physical information. If you have other partitions on the same physical disk, dragging any one of them into the Analysis window will fill in these parameters automatically. Press the
Analyze
button
to start a scan of the disk. DiskSalv will construct an internal representation of any volume it finds. These will show up by the volume name of the partition as found. DiskSalv does not support all modes on such partitions, as some are only meaningful when DOS is active on the partitions. In many cases, only the RDB itself is damaged. DiskSalv can write any partition data to disk in Mount form with the
Save Device
function, or to the RDB with the
Save to RDB
function.

Such files can be mounted with the AmigaDOS Mount command with little or no modification. Some Rigid Disk Block editors, such as RDPrep from MicroBotics, Inc. can convert between mountlist and RDBs. The next release of DiskSalv 3 will also allow this to be written directly to RDB.

1.33 giveitaway

The DiskSalv 3 program is copyrighted, commercially distributed software, and absolutely must not be given away, copied, modified, or otherwise mistreated. The success of DiskSalv 3 today determines the future of the program.

However, as long as it will reasonably fit, the DiskSalv 3 distribution disk will contains the current DiskSalv 2 distribution, in archived form. DiskSalv 2 may be given away, uploaded, copied, etc. on a not-for-profit basis. In fact, I encourage this. Only the materials in the DiskSalv2

directory may be freely redistributed.

1.34 2304

Immediately after the introduction window, DiskSalv's Input Window will come up. This is the main DiskSalv window. Every DiskSalv operation will ultimately return here when complete, unless the user chooses to quit DiskSalv. As the name implies, this window manages forms of input to DiskSalv. The user must specify at least two things to DiskSalv: an input device, and a mode of operation. Options may be selected, but are not necessary in most cases. Once the input setup is compete, the disk salvage operation begins with a press of the Scan button.

Various secondary operations can be launched from the input window via menu items or function buttons. There are a number of these secondary functions available, most of which support options related to the Major Modes or several kinds of device management.

Topics:

Device Setup

Major Mode

Button Options

Project Menu

Settings Menu

1.35 devicesetup

There are three components to a DiskSalv device selection. The primary component is the device itself. Every device has one of the AmigaDOS file system format types, which must be correctly entered. Finally, many modes support a pattern, which can be used to place a set of constraints on which disk objects will be processed by DiskSalv during the selected operation.

Topics:

Device Selection

File System Selection

Pattern Selection

1.36 2200

An Input Device must be selected for any
Major Mode
operation. The
Scan button will be ghosted until such a device has been selected. Device selection can be made simply by dragging and dropping a disk icon into DiskSalv's
input window
. Alternately, the user can click on the "Device:" gadget. This brings up a list requester containing all valid device options.

The selected
Major Mode
will determine just which devices are valid for a particular operation. It is important to know a bit about how devices work on the Amiga to understand what shows up here. The AmigaOS provides several levels of device abstraction, as shown here. The Exec-level interface is very simple, supporting just the few commands needed to read and write basic data to and from a variety of hardware types in a device-independent fashion. The DOS-level interface is more sophisticated, supporting the knowledge of files and directories.

Since a disk error is inherently a problem the given AmigaDOS file system can't resolve, DiskSalv must operate at the Exec level to run most of its operations. Thus, in the diagram above, DiskSalv could not process the RAM: disk, since that device's file system talks directly to the hardware (memory, in this case). The operations performed by DiskSalv are based on its knowledge of the AmigaDOS file system structure. Since the CD0: device above uses the CDFileSystem, which addresses the ISO9660 file format rather than the Amiga file format, DiskSalv can not process CD0: either. It can process DH0: and DF0:. A variety of checks are made on each device in the system to determine DiskSalv suitability, depending on the selected mode.

Topics:

Basic List Views

Device List Requester

Read New Devices

1.37 basiclistviews

The Device: gadget is typical of pop-up list requesters used in ↩
DiskSalv.
Similar list requesters are available for FileSystem and
Pattern
selection on the
input window
, and for other selections elsewhere in the DiskSalv program. In all such requesters, a list is displayed which scrolls in realtime based on movement of the proportional slider or

clicking of the arrows. In some cases, the list view provides a second slider and set of arrows for scrolling on the horizontal axis. A current object is indicated with backfill, and in some cases multiple selections are shown in highlight. Unlike the gadtools list view used by many programs, the DiskSalv list view behaves the same in all version of the Amiga operating system.

1.38 devlistreq

The Device List requester lists the AmigaDOS device name of each device in
the system. It will only list devices that are suitable to the selected

Major Mode
in effect at the moment, in this case Salvage mode. Next each device name is the logical volume name of the disk in that device, if any, since Salvage mode uses physical devices. If Backup mode were selected, this requester would list logical names first, and the physcial device associated with each, if any, as the secondary name. A device is selected by clicking on a name with the mouse, then clicking the Ok button. Alternaterly, a double-click on the name will select it. A click of the Cancel button will leave the Device List requester without making any selection.

1.39 2213

A final feature of the
Device List
requester (though not found in other
list views, naturally), is the single menu item, Read New. Selecting this menu item causes DiskSalv to reinitialize its device list from the global AmigaDOS device list. If any devices are added via Mount or other means after DiskSalv starts, this function will pick them up.

1.40 2201

When an input device is selected, DiskSalv attempts to determine which of
the Amiga file system types is in use on the device. Ordinarily, this can be done simply, by reading the root block of the device's partition, the standard place to store the file system type code. When DiskSalv finds a meaningful code, it will automatically select that file system for processing. In most cases, the user doesn't do anything here.

However, there are some times when user intervention may be called for. When it is, the file system type can be changed by clicking on the File System: list request gadget. This will pop up a list of the file system types known to DiskSalv. It is extremely important to select the proper file system for a device. If the wong file system is selected, the file system structure on that disk can be permanently damaged!

There are a few times the user may wish to override the file system type selected by DiskSalv. It is possible that a disk crash caused the file system type stored on-disk to be incorrect. This is most common when a disk has been accidently formatted. For example, I might have a hard disk formatted as Fast FileSystem, but perhaps its default type is Original FileSystem. Let's say I accidently type:

```
1> Format DEVICE DH0: NAME "Whoops!" QUICK OFS
```

DiskSalv can completely reverse this operation using the Unformat mode. But it will think the partition is formatted with the Original File System Not only won't that work, but it would cause DiskSalv to corrupt this input disk's structure. That's how important the proper file system selection is.

When the file system is unknown, the user's best recourse is to select the

Best-Guess
file system type. This is not actually a file system, but a special pseudo file system. When selected, DiskSalv will run extra analysis steps during its scan of the input device, which are then used to decide the type of the disk. This isn't perfect, for several reasons. The first reason is, of course, that DiskSalv at present knows just these file systems:

```
OFS   Original File System.
FFS   Fast File System.
OFS Intl. OFS with ISO 8-bit character support.
FFS Intl. FFS with ISO 8-bit character support.
DC-OFS    OFS with directory caching
DC-FFS    FFS with directory caching
```

These are the file systems that DiskSalv can process. It attempts to reject any other file system. In some cases, a device can be detected as some kind of custom file system, such as RAM: or CD-ROM (ISO9660 file system). When these are detected, no

device list
entry is made for them. In other cases, the device type can be detected as a type not supported by DiskSalv, or an unknown type. Examples of these are:

```
FAT   MS-DOS File System
FAT12   Another MS-DOS File System
NDOS    A disk specifically marked as Not a DOS Disk
COPY    A diskcopy failed on this disk.
```

DiskSalv tries to make an intelligent guess about the file system type recorded on the disk. If the type is unknown but seems to be a well formed file system code (eg, it might be a valid, unknown file system), DiskSalv will keep the unknown file system code for internal use. It will display the code, but it can't do anything to the disk (and will prevent the user from doing so). When no probable type can be determined, the default file system is selected. This defaults to

Best-Guess
, but can be changed via the

DEFAULTFS

command parameter.

1.41 bestguess

The Best-Guess file system is actually a pseudo file system supported by
DiskSalv. It causes DiskSalv to process a scan with no preconceived notion of the file system type, and to keep track of statistics on the format as found. In most cases, this can automatically determine the type of the underlying file system, assuming it is one of the standard AmigaDOS formats, of course.

Best-Guess can also be selected to process a damaged disk, perhaps a COPY disk, but it makes no sense on MS-DOS disks. Also note that the low-level disk format is a factor. This is controlled by the Exec-level device driver used. So DiskSalv can determine the identity of an MS-DOS disk through PC0:, but not through DF0:, since the low-level format is the MS-DOS type as well as the file system format.

The Best-Guess mechanism works well on damaged AmigaDOS disks. It can, however, be confused by the history of the disk. For example, consider a full OFS disk that's reformatted to FFS, then filled rather sparsely. There's an excellent chance, at this point, that Best-Guess would detect this as an OFS disk, since based on the disk's content, it is more than it isn't. There wasn't much that could be done about this in the past. Nowadays, it's a good idea to run
Cleanup
mode on a freshly formatted
disk, once it's certain that this new format was a good idea.
Cleanup
will eliminate any trace of the previous format, even if the new format was
put in place with a Format Quick command.

1.42 2215

A final option of the Device setup is the selection of an optional

pattern
. A pattern can be selected to include, exclude, or search for an object or set of objects being scanned. DiskSalv maintains any number of named pattern sets, which are selected by name by clicking on the Pattern: button. New
patterns
can be entered by editing a pattern file and loading
it with the
Pattern Load
funcion.

1.43 2202

```
                The Major Mode defines the operation that will be run on the  ↩
                   selected input
device during when the Scan button is pressed. This is set via a cycle
gadget on the bottom left of the
                input window
                , directly below a graphic
that depicts the selected major mode. The setting of the major mode will
have an effect on the options offered in the
                input window
                , such as device
type and
                pattern
                 support.

  Topics:

                Salvage

                Undelete

                Repair

                Unformat

                Check

                Backup

                Cleanup
```

1.44 mmsalvage

```
                The Salvage mode is the default mode when the
                input window
                 first comes
up. This is essentially the mode of last resort. In Salvage, the disk is
scanned for any objects that can be found. These objects must be
recongnizable to the scanner as valid objects, but may not be understood by
any Amiga file system. DiskSalv uses a set of pattern matching routines
based on fuzzy logic to determine if a given disk block could contain
something recognizable enough to recover.

Once the disk has been scanned, the scanner window changes into the

                output window
                . In this window, the user can select objects to recover.
Objects are not changed on-disk, but are in fact copied over to an
alternate volume of some kind. This operation is called
                recover-by-copy
                .

There are a couple of things to appreciate about the Salvage mode. Since
DiskSalv can find anything that's on the disk, it will often find unwanted
or incomplete files. Normally this is no problem, but each file scanned
```

requires a small amount of memory, and of course each file selected for restoration requires space on the selected output device. One may use the

pattern
mechanism to define patterns to break a scan up into pieces, exclude files that are otherwise backed up, etc.

Undeleted files are generally restored in full, though a damaged file, perhaps the cause of the disk crash in the first place, may be restored only in part. This is also true of deleted files that have had part of their contents reallocated. DiskSalv makes no attempt to decide whether such files are useful or not. It will by default set AmigaDOS file notes on any file that appears damaged in some way. Files that are severely damaged may be difficult to restore in much completeness, but files with only slight damage can often be restored in near totality.

The Amiga's Original File System provides good redundancy on the contents of a file, while the Fast File System does not. Therefore, more can usually be recovered from a crashed OFS disk, all else being equal. This is primarily due to extra accounting information stored along with every OFS data block. FFS data blocks store only data. This is also why FFS is faster and stores 6.7% more data per block, on partitions that use 512 bytes per block. For extra security, it is a reasonable idea to use an OFS partition. The efficiency of OFS goes up with increased block sizes, which are available globally in AmigaDOS 2.x and above (by adjusting the bytes per sector setting), or on a partition by partition basis in AmigaDOS 3.1 and above (by adjusting the sectors per block setting).

Unlike a file or link, a directory does not have to exist on-disk to be recovered in Salvage mode. The existence of a directory can be determined if any of its children is discovered on-disk. DiskSalv uses this mechanism throughout the scanning process. When a file is discovered, it is placed in a directory. If that directory hasn't been found yet, DiskSalv will create a temporary directory for it in the DiskSalv_Extras directory for the output disk. When the scan has completed, any objects with missing directories will be found in their temporary directories. Thus, DiskSalv can get back a whole disk's contents, in the original tree structure, even if every directory on the disk were somehow eliminated.

The Salvage mode is the most forgiving DiskSalv mode when it comes to disk integrity. There is no requirement that a disk's bitmap, root block, or much of anything else be intact on the disk. DiskSalv will attempt to adjust for an incorrect partition definition on a file-by-file basis. However, if a partition is off by much, some file will not be found. Users of DiskSalv 1.x should recognize the Salvage mode as essentially what this early version of DiskSalv did. While the intent is the same, the modern version of Salvage works considerably better than the original DiskSalv version of the function. The Salvage mode of DiskSalv 2 is the same function.

1.45 mmundelete

The Undelete mode is designed to recover accidently deleted files. ↩
Unlike
most other DiskSalv modes, this function will only operate on a good

device, one that is validated under AmigaDOS. It uses the disk's bitmap to determine which disk blocks are unused by AmigaDOS, and scans only those blocks.

As in the Salvage mode, this is a recover-by-copy mode. When the disk scan has completed, the user is presented with a list of directories and files in the
output window
file browser. Selected files can be copied to any other AmigaDOS disk volume or pipe-like object, such as a
TAPE:
device.

Like Salvage, the Undelete mode supports DiskSalv
patterns
. A pattern can be constructed in any text editor, several useful defaults are included in the "DiskSalv.pattern" file. This must be done a bit more carefully that when dealing with undeleted files, however. It's not only possible, but quite common, that while a file still exists, its parent directory block has been reallocated.

1.46 mmrepair

The primary
fix-in-place
mode is the Repair mode. This is designed to analyze an AmigaDOS device in-place and correct it such that its structure is a legal one for mounting under its an AmigaDOS file system. Ideally, the Repair mode will restore a disk to full read/write operation with no data loss. In practice, however, it's often necessary to remove an object or two in order to repair the disk's structure. Files can become damaged for a variety of reasons. DiskSalv does not attempt to repair the on-disk structure of a file. If a problem file is encountered, it will be de-linked from its parent directory on-disk, and presented for
recover-by-copy
on
the
output window
after the
fix-in-place
operation has completed.

Just like files, directories can be found damaged on a disk. However, DiskSalv does attempt to reconstruct a damaged directory. Based on its full scan of the input device, DiskSalv's internal disk model will have a record of the name, if available, and the contents of any directory either found directly or implied by a child's reference. With this, DiskSalv can reconstruct a damaged directory in full. Only a hard error will prevent this form of recovery and require a directory to be eliminated from the disk's directory tree.

There are two variations on the Repair mode: slow and fast. The slow mode is indicated by setting the scanning speed option button to slow . This causes DiskSalv to scan the entire disk from start to finish, which takes

awhile, but builds a very complete model of the disk. When the scanning speed option button is set to fast , the scan can be completed in much less time, by walking the disk's file structure from the root block on down. This builds a less complete model of the disk, and does not deal with damaged directories very well. It can solve simpler problems, though, and it's much faster than the slow version of Repair mode, especially on large disks. The fast Repair never undeletes files, while the slow Repair can undelete files, but only in the process of reconstructing a directory from scratch. The intent here is to fix the disk's structure, not undelete anything the user has specifically deleted.

When the Repair mode has finished, it leaves the input disk with its bitmap set invalid. This causes the Amiga's file system validator to process the disk, a perfectly normal behavior. The philosophy behind this is simple -- the file system is the ultimate judge of the correctness of a disk, and it has a routine (the validator) to determine this correctness. It would be both unwise and wasteful for DiskSalv to attempt this final pass.

Unlike the
recover-by-copy
modes, the Repair mode (and other
fix-in-place
functions) can not use a DiskSalv pattern to filter files during ↩
scan. The
ability of a Fix-In-Place run to repair the input disk is directly tied to the completeness of its disk model. Any exclusions would compromise this model.

1.47 mmunformat

The Unformat mode is a
fix-in-place
designed to reverse the accidental
formatting of a disk. In truth, if a disk is formatted in full, there is no going back. A full format in most file systems erases all data on the formatted disk.

Yet a format isn't always full. The Format command itself supports the Quick option. This causes the disk's file system type to be updated, and it writes an empty root block. So, to the user, the disk is clean, but in fact, very little is destroyed. DiskSalv can get everything back when this is done accidently. Alternately, the user may abort a full format before it is complete. DiskSalv may be able to partially restore such a disk in-place. If not, the Salvage mode can get back anything that's still viable.

Care must be taken to set the file system type properly. The user must make absolutely certain that the file system input to DiskSalv is the type of the format being recovered, which is not necessarily the type of the new format recorded on disk. If there is any doubt, use the
Best-Guess
pseudo
file system type. This will analyze the format on-disk, and select the type that best matches this format. This works very well if there has been only one format on the disk, or if the disk was very full. It can fail if the

number of objects from an even older, different format outweigh the number of active objects on the disk structure being restored.

The Unformat mode is not only useful for in the case of an errant format. The scanning and reconstruction mechanism is exactly the same as the slow variation of the Repair mode, with a slightly different emphasis. During a Repair, every directory found is analyzed for correctness on-disk. If it is in anyway damaged, it is rebuilt from DiskSalv's internal tree model. Under Unformat, every directory encountered is rebuilt from DiskSalv's model. This has the effect of bringing back any file that is still viable. So in a way, Unformat can be thought of as a global Undelete-in-Place, and it may be used as this. The main disadvantage of such an operation is that may bring back a large number of old files that no longer have any reason for being.

1.48 mmcheck

The Check mode is a fix-in-place mode that doesn't actually fix anything. Instead, it reports what a Repair mode run probably would have done. It is useful to check suspect disks before running DiskSalv to repair them.

As implied, the report generated by a check run can not always be an exact one. During a real fix-in-place run, repairs made early in a run can affect things done later in the run. In general, though, the check report is a worst-case report, and since it does not change anything on disk, it's always safe to use.

1.49 mmbackup

As a preventative measure, DiskSalv offers the Backup mode. Unlike all other modes, Backup operates on logical volumes rather than physical AmigaDOS devices based on the AmigaDOS file systems and standard Exec-level device drivers. Thus, DiskSalv can backup from any kind of file-oriented AmigaDOS device (disk, network, CD-ROM, etc.), even a logical device created with the AmigaDOS Assign command.

This mode scans the given logical device by walking its file structure and building a standard DiskSalv tree model. Like any recover-by-copy mode, this scan can be altered by a complex pattern. It's very common to exclude any files that have the archival bit set, though this is of course just an option. It is also reasonable to use a pattern that matches any file that doesn't need to be backed up. Such files may include easily installed commercial software (it's already on floppy or CD-ROM on your

bookshelf) or various kinds of output files (many, though not necessarily all, object or postscript files, for example, are easily recreated from their sources).

Optionally, the archive bit may be set on each file that DiskSalv backs up. This can be set via the Settings menu on the
input window
. While the effect of this does not take place until actual output is done, it is a parameter affecting the input disk and, therefore, addressed on the

input window
.

Once a scan is complete, the
output window
 is called up just like in a
recover-by-copy
 operation. This allows the selection of files encountered during a scan, and it allows the output device to be selected. Just as in the other Recover-by-Copy modes, backup can be made in file structure format to any AmigaDOS disk-oriented device. Alternately, it may send the backup set out in DiskSalv's Archive Stream format to any pipe-like device, such as a file, a PIPE: device, or a
TAPE:
 device.
By linking together pipes, any external compression protocol that supports piping may be used to compress the DiskSalv stream before it goes to your choice of media. DiskSalv does not currently provide any automatic piping mechanism, though it will in the future.

1.50 mmcleanup

The Cleanup mode is a disk maintenance option. Like other modes, ↩
this scans an input device. However, rather than looking for things to fix, it looks for things to eliminate. After months or years of use, a disk, especially a large hard disk, can have quite an array of deleted files in various states of disrepair somewhere on it.

Cleanup is designed to locate these and eliminate them for good, by erasing them on-disk. This has a couple of uses. Since it simplifies the disk's history, it makes Salvage and Undelete modes easier to work with, since fewer garbage files need be sorted through if a salvage is needed later. This also improves the reliability of a future
Best-Guess
 scan, since any previously existing formats are wiped away when this is run. This mode is also useful at cleaning any possible unwanted deleted files from software release disks. While uncommon today, early DiskSalv users reported finding smatterings of various program sources on early release disks from several companies.

The Cleanup mode has some restrictions. Like most other DiskSalv functions,

it can only operate on a physically based AmigaDOS device with one of the aforementioned file system types on it. Additionally, it requires the input disk to be fully validated, both as seen on-disk and as reported by AmigaDOS. This is because it bases its scan of unused blocks on a device's bitmap. If the bitmap isn't valid, DiskSalv would probably damage the input device if not checking carefully.

1.51 inputbuttons

DiskSalv provides a set of useful functions on a row of button gadgets below the device display on the input window. None of these functions are required for the proper operation of DiskSalv, though most of them are quite useful.

Button Options:

Information

About...

Pattern Selector

Scanning Speed

Log File

Restore Stream

Device Editor

Device Analysis

Load Device

Save Device

1.52 2204

The Information button displays additional details about the selected device. This information, which varies according to device type, is rarely very useful to the user, but it is provided anyway. AmigaDOS supports four kinds of devices, from its point of view.

A physical device represents a normal AmigaDOS device description constructed from the AmigaDOS device environment list by DiskSalv. A physical device is fully initialized by DiskSalv, and can be used in any DiskSalv mode. A slight variation is the unmounted device. This indicates a full-featured AmigaDOS device, complete with file system and all, that has

not yet been initialized by AmigaDOS. It is common for AmigaDOS device to be physically initialized on-demand, when first used. The unmounted designation allows DiskSalv to keep them in this state. AmigaDOS functions are not used on unmounted devices by DiskSalv, since this will cause them to be initialized by AmigaDOS.

The third kind of device is the virtual device. Any device description not supplied to DiskSalv from AmigaDOS is labeled virtual. These include partitions found during an Analysis run, descriptions entered by hand in the

Device Editor
, or descriptions loaded from
DOSDrivers
files.

DiskSalv will not use any AmigaDOS functions on a virtual device.

The final device type is the volume. A volume is any AmigaDOS device, volume, or assignment. At present, volumes only exist in Backup mode, and they are the only device type supported there. If a device is selected in any other mode, it will be converted to a volume, if possible, when Backup mode is selected. If Backup mode is left for another mode, any volume device selected will be converted to a physical device, if possible. If the device conversion can not be done, the device entry will be cleared.

An example of the Device Information requester is shown above. All physical, unmounted, and virtual devices are displayed something like this, while the volume display contains much less data. If the Device Information display shows up as a tall, thin column of text that extends off your screen, you have a problem. There are several Requester Improver programs out in the freely redistributable software channels. These seeks to generate fancier requesters by replacing the AutoRequest functions in Intuition. When a such a program’s replacement function doesn’t properly support the features of Intuition’s function, you may see this distortion in some DiskSalv requesters. DiskSalv is doing nothing wrong, the fault here lies with the replacement AutoRequest function. Eliminating the Requester Improver, or using one that properly emulates Intuition, will fix this problem for you.

1.53 inputbuttons.about

This button displays information about the program and the author. It displays the release and internal version number of DiskSalv 3. This information is a critical debugging aid when any problems are reported.

1.54 2203

This button calls up a file requester, to load a
Pattern
file. Any

number of named complex patterns may be included in a pattern file. A complex pattern can match, exclude, or search for specific files and/or directories. Patterns allow matching against files, directories, or both,

file notes, protection, date, etc. The pattern to be used for a scan is selected via the
Pattern Select
button.

1.55 2214

The Scanning Speed button is a toggle button that selects between slow and fast disk scanning algorithms. At present, only the
Repair
mode has two possible scanning algorithms, so this button is disabled in other modes. As you might expect, there is a tradeoff between the speed at which a disk can be processed and the throughness of the processing. When set for slow scanning, DiskSalv looks at every block on the input device to build its tree model for Repair. When set for fast scanning, DiskSalv walks the tree structure of the input device instead. This tree walk is generally much faster than a complete scan, but it's not quite as complete. Generally, the fast Repair (called Validate in DiskSalv 2) can fix minor problems just as well as the slow Repair. Slow Repair is recommended for severe problems, and actually required if DiskSalv can't make sense of a disk's root block.

1.56 220d

This calls up a standard Amiga file requester to create a Log File. Everything that happens in any DiskSalv Scan window will be recorded in a log file. All of the important results of any DiskSalv scan, including the modeling scan, backups, analysis, file recovery, or archive restoration is logged in this file. Every event that takes place in the Scan window is tagged with a scan operation code, making it easy to search though a log file run on even extremely large disks. Obviously, a log file must not be created on the input device, but it can be created on any logical AmigaDOS device, including SER:, PAR:, or PRT:.

1.57 2208

This calls up the Stream Restoration requester. As described in chapter 6, the results of a Backup or Recover-by-Copy operation can be sent to an archival stream rather than an AmigaDOS file system. DiskSalv manages this stream format, which preserves the tree structure and contents of the device being processed, but writes it out as a single object. This can be sent to a pipe or a file.

DiskSalv streams may be real useful when copy out files from a disk, but they're of little use until they're restored. The Stream Restoration requester accepts the name of a DiskSalv stream and some AmigaDOS file-structured device for output. They are entered into string gadgets, but standard file requesters may be brought up for either one by pressing

the file requester button associated with each string gadget.

Once the fields have been filled in, the Start button is enabled. A press of this will open the DiskSalv
Scan
window and start the restoration process. As with all Scan operations, progress is indicated by tallies of the objects found. Results are displayed line by line as they occur (this is the same data written to a log file).

1.58 2212

A press of the Device Editor button will change the input window into the
Device Editor
window. The device editor provides a simple way to enter a new device description, or edit an existing one, from within DiskSalv. Any device so entered becomes a virtual device in DiskSalv terms.

There are several ways to use this. If a device is currently selected on the
input window
, it will be automatically entered into the Device Editor. Any device on the Amiga workbench dropped into the Device Editor window will be entered in place of the current device, if any. After edits are made, the
input window
is restored by clicking on either the
Create
button, to keep the device, or the
Cancel
button, to return with no changes made. DiskSalv checks any newly entered device against existing devices, and will only permit unique devices to be created.

1.59 2205

The Device Analysis button calls up the
Device Editor
in its device analysis mode. This mode is designed to search an entire physical disk for any AmigaDOS partitions that may exist on it. The search can be directed to find a particular volume, or all viable volumes on the disk. Data for the search device is set up in the Device Editor much like for a traditional device edit. If a disk icon is dropped into the window at this point, the device editor will attempt to figure out the full size of the physical disk that partition is on by looking for any other mounted partitions on the same disk. Once the information is set up, click the
Search
button to

do search by pattern for a specific volume,
Analyze
to find as many
volumes on disk as possible, or
Cancel
to go back to the
input window
.

Pressing either of the former buttons will start up the DiskSalv
Scan
window. In this mode, only partitions are of interest. A tally of partitions and errors is kept, and anything found is displayed. A continuous report of errors generally indicates the scan has run off the end of the disk. Once the scan is complete, the
input window
is
restarted. The
device list
will contain any new device descriptions discovered during the scan.

1.60 220f

This button brings up a standard file requester, to allow an ASCII ↩
device
description file to be loaded as a virtual device into DiskSalv. Such a file must be in the
DOSDrivers
file format. A DOSDrivers file contains the description of a single device, using the same notation as originally defined for the system-wide MountList file. DOSDrivers files have been preferred on the Amiga, rather than the single MountList, since AmigaOS 2.00. Rather than use the Load Device button, a
DOSDrivers
file may simply
be dropped into the
input window
.

1.61 220e

The inverse of Load Device, the Save Device button brings up a ↩
standard
file requester to allow the currently selected device to be written out to disk in
DOSDrivers
format. If no device is selected, this button is disabled. Any device, including those found during an Analyze run, may be written out to disk. Thus, a missing AmigaDOS device may be located with Analyze, written to disk, and mounted under AmigaDOS via the Mount command. DiskSalv provides everything necessary in these files to physically

describe the partition. Extra such as Mask, FileSystem, MaxTransfer,
GlobVec, etc. may have to be added by hand to support the mount properly.

1.62 inputproject

The Project menu is largely redundant in DiskSalv 3. Most of its options,
originally defined for DiskSalv 2, are available via function buttons or
other gadgets. It is retained in any case for completeness.

Menu Options:

About...

Help...

Log File...

Restore...

Quit

1.63 2207

The Quit item, like the close gadget, unconditionally quits DiskSalv. There
is no difference between the two, though some users prefer one method over
the other.

1.64 inputsettings

The Settings Menu offers a number of optional settings, and the option to
save the settings made here. These settings adjust things that happen to
the input device or during the forthcoming Scan operation.

Menu Options:

DOS Lock

Low Memory

Small Window

Quick Scan

Set Archival Bit

Internal Help

Save Settings

1.65 2209

Once the Scan starts, DiskSalv will inhibit a physical input device, effectively shutting down the file system on that device. This prevents AmigaDOS from doing anything to the disk during DiskSalv's run, which is the proper thing to do according to the AmigaDOS specifications. However, on occasion this causes problems. For example, it's possible, especially for Undelete runs, that DiskSalv is being used on the SYS: disk. Shutting down the SYS: disk is not generally a good idea. When unchecked, no inhibit will be used on the input disk. When running in this mode, it is very strongly suggested that all other programs that be shut down are shut down. A write to a disk being examined by DiskSalv can cause incorrect results. Because of this, DiskSalv always inhibits when performing a fix-in-place operation. To run a fix-in-place on the SYS: disk, boot up with a different SYS: disk. This corresponds to the KEEPDOS command parameter.

1.66 220a

While DiskSalv is not at all wasteful with memory, it does use memory here and there to improve the speed of disk processing. If memory is really tight on a system, checking this option may save enough to let DiskSalv do its work, at the expense of extra time. Of course, in such cases, all extra software should be shut down. It's still possible that some systems will have disks that are too large to process in the memory available.

Patterns can be used to break a Salvage run up into several pieces, by restricting the scan in various ways. All fix-in-place operations must occur in one piece. This corresponds to the LOWMEM command paramater.

1.67 2210

Ordinarily DiskSalv will adjust the Scan window according to the size of the screen DiskSalv opens on. If the screen and font size permit, a reasonably large Scan window is opened, leaving room for a good sized Results display. Checking this menu option will prevent this, keeping the window as small as possible. This corresponds to the SMALLWINDOW

command paramater.

1.68 2211

Checking this menu option will increase the speed of a disk scan by eliminating the Results display from the Scan window. Depending on the screen type and hard disk speed, the elimination of this text display and window scrolling can significantly increase the speed of the scanning process. If a log file has been selected, it will still get the result, but of course that will defeat the purpose. This corresponds to the QUICKSCAN command parameter.

1.69 2216

AmigaDOS file systems support an Archival bit, a bit in a file's protection field that indicates that the file has been backed up. Checking this item will cause DiskSalv to set the archival bit on the input disk of any file it backs up. At present, this feature is only enabled in Backup mode. This corresponds to the NOARCHIVE command parameter.

1.70 220b

Ordinarily, DiskSalv will call up AmigaGuide as a help server if it can find an appropriate DiskSalv.guide file. If, for some reason, AmigaGuide help is not desired, checking this option will cause the internal help text to be used instead in response to any help events. This option is also of use to users of certain V34.x releases of AmigaGuide.library. For unknown reasons, this library crashes when called up by DiskSalv (or any other program) as an asynchronous help server. The AmigaGuide help can be turned off on startup via the NOGUIDE command paramater. However, upgrading to a corrected release of AmigaGuide.library will enable the AmigaGuide help server, which is far superior to the internal help.

1.71 220c

Selecting this item will cause DiskSalv to write the state of each of these items to the DiskSalv icon. Each item is controlled by a command parameter (Icon tooltype or shell-based command-line option). These may be set manually on the shell's command line or by editing the DiskSalv icon via the Workbench Information... function.

1.72 2206

Once the input paramaters have been set up to DiskSalv's satisfaction, the Scan button is unghosted. Clicking on this device will start the disk scan. The input window will be replaced by the Disk Scanner.

1.73 patterns

DiskSalv supports a complex pattern matching mechanism, which can be used to control which files are scanned by DiskSalv. Patterns are considered when running in Salvage, Undelete, or Backup modes, ignored in all other modes. By default, DiskSalv reads patterns in from the DiskSalv.pattern file, if present.

Patterns are defined in a C-Language-like syntax. There are two main types of patterns: patterns and groups. A pattern handles a single set of matching attributes, while a group can contain any number of patterns. Patterns and groups are always named, and can be selected by name in the pattern selection function.

Types:

pattern

group
Attributes:

path

note

```
                date

                size

                protection

                match
                  Miscellaneous:

                comments
                Complex patterns can be edited with any ASCII text editor. Any  ↩
                   number of
files can be loaded into DiskSalv. Loads are additive -- a load doesn't
overwrite the existing patterns.
```

1.74 pat.names

```
A pattern name may contain any ASCII characters except SPC, TAB, LF, CR,
VT, {, or }. Ideally, a name is descriptive of the pattern's function.
Spaces may be embedded in the name as long as the name is enclosed in
quotes.
```

1.75 pat.attributes

```
                Patterns may contain one of each of the following attributes:


                path

                note

                date

                size

                protection

                match
                When an attribute is not supplied, the associated item is not  ↩
                   considered
when a pattern is matched against a file or directory.
```

1.76 pat.compare

```
Comparisons are done using the standard comparison operators used in most
computer languages:

    = Match file properties which are exactly equal to the item.
    > Match file properties greater than the item.
```

```
    < Match file properties less than the item.
```

1.77 pat.pattern

```
                A pattern is specified in C-like syntax:

    pattern
                <pattern_name>
                 {

                <attributes>
                    };

The pattern can be referenced by name in the
                pattern selection
                list.
```

1.78 pat.group

```
                A group is specified in C-like syntax:

    group
                pattern_name
                 {

                patterns
                    };

When multiple patterns match the same file or directory, the first pattern
listed takes precedence over any others.
```

1.79 pat.path

```
                The path() attribute matches a file or directory name. The syntax ↩
                   is:

    path(
                AmigaDOS pattern
                 [,"any|file|directory}"]);

The "any" qualifier matches against files or directories, and it is the
default. The "file" qualifier matches only files. The "directory" qualifier
matches only directories.
```

1.80 pat.note

```
                The note() attribute matches against a filenote, as set via the ↩
                   AmigaDOS
FileNote field of a file or directory. A standard AmigaDOS regular
expression is used for the matching. The syntax is:

    note(
                AmigaDOS pattern
                );
```

1.81 pat.date

```
                The date() attribute matches against the date stamp on a file or ↩
                   directory.
The attribute can specify a
                comparison
                 and date, using a standard

                AmigaDOS date
                 specification. The syntax is:

    date(
                <comparison>
                ,
                <AmigaDOS date>
                );
```

The comparison considers only the date, not the actual time of day of a file.

1.82 pat.size

```
                The size() attribute matches against the byte size of a file. The
attributes can specify a comparator and a file size. The size may be bytes,
Kilobytes (K), Megabytes (M), or Gigabytes (G). The syntax is:

    size{
                <comparison>
                ,"size");
```

Directories are considered to have a size of zero bytes.

1.83 pat.protect

File and directory pattern bit comparisons can be specified with the protect() directive. Each protection bit can be matched set, matched clear, or ignored. The syntax is:

```
    protection{"<protection bit string>");
```

```
Where a protection bit string consists of the following:

    D     Match the Delete bit set
    E     Match the Execute bit set
    W     Match the Write bit set
    R     Match the Read bit set
    A     Match the Archival bit set
    P   Match the Pure bit set
    S     Match the Script bit set
    !<PB> Match the following protection bit clear

Some useful protection attributes:

    protection("!A"); Matches unarchived files/directories
    protection("E");  Matches only files set as executable
```

1.84 pat.match

```
The match() attribute decides the action that should be taken for anything
that matches the rest of the complex pattern. The syntax is:

    match("include|exclude|stop");

The result of the actions is:

    include The matched item is included in the scan
    exclude The matched item is exclued from the scan
    stop  The scan stops on a match, the item is included
```

1.85 pat.comment

```
C-like comment blocks may be included anywhere in a pattern file. They are
completely ignored by DiskSalv. The syntax:

    /* This is a comment */

Comments can be nested, but the start and stop tokens must be matched.
```

1.86 deviceedit

```
                The Device Editor is started from the
                input window
                , by clicking on
either the Device Editor or Device Analysis buttons. This window
replaces the input Window.

  Topics:

                Device Selection
```

Device Edit/Creation

Device Analysis

Rigid Disk Block Functions

Parameter Fields

Menu Items

1.87 deviceselect

While a device description can be entered into the Device Editor completely
by hand, it is more often the case that an existing description will be the basis for a new device or an analysis pass. There are several ways to enter a device's decription into the editor.

Topics:

Current Device

Workbench Device

DOSDrivers File

1.88 devsel.current

In truth, it's rather unusual to enter the Device Editor without some
device decription automatically brought in, though it is possible. If there is no current device selected in the input window, the Device Editor
will start up with all of its data fields either empty or set to program default values. If there is a current device selected, the Device Editor
starts up with its physical parameters entered.

1.89 devsel.workbench

Once in the Device Editor, any AmigaDOS device can be dropped into
the window. If the device is suitable for DiskSalv, it will be entered into the Device Editor. Any previous device description is discarded. If the device is not suitable, it will be rejected, and the display will

flash.

1.90 devsel.dosdrivers

Drag and Drop

A
DOSDrivers
file with an appropriate device description can also be dropped into the Device Editor. If the
DOSDrivers
file contains the proper format and is suitable to DiskSalv, it is entered into the Device Editor. If there are any problems in format of or specification generated by the
DOSDrivers
file, it will be rejected, and the display will flash.

Load by Filename

The "Project/Load from File..." menu item will load a new file description by name, rather than dropping. This command brings up a standard AmigaDOS file requester which can be used to get the file.

Save to File

A device description can be saved to disk from within the Device Editor by selecting the "Project/Save to File..." function. This brings up a standard file requester, lets the user specify the name of the output file.

1.91 7306

The Device Editor lets the user enter a new device description or ↩
alter an existing one. Any of the device
parameters
may be modified to make a new device. The Device Editor will not allow a device to be created until every required field contains something reasonable. Once the device has been edited to the user's satisfaction, it can be finalized by pressing the

Create
button. The editor works in temporary storage, so that the user can back out from the edit by instead pressing the
Cancel
button if there is any problem.

Note that while the Device Editor does checking on the specified device description, it does not actually try it out. Thus, it is possible for a device to be entered into the

device list
 that's not properly formed.
An attempt to open such a device, however, will be reported by DiskSalv as an error. Any time a device open fails like that, it is removed from the device list.

1.92 720a

When the Create button is selected, several things happen. ↩
 DiskSalv stores
devices by AmigaDOS device name, and will only store one device under any name. A warning requester will come up if the name of the device to create is the same as an existing device, allowing the user to replace the existing device or return to the
Device Editor
. A warning will also be
issued if the device description is physically identical to a device already in the
Device List
.

1.93 720b

The Search button starts up the device analyzer, based on the ↩
 device
information entered in the
Device Editor
. The Device Editor will keep
this button ghosted until enough data has been entered to make such a scan possible. When a search run is selected, the scanner will start up and display any scanning progress. An internal device model will be constructed for any the first volume found during the scan that matches the required simple AmigaDOS pattern. This device, which will be entered in the

Device List
 by volume name, can be saved to disk via the
Save Device
button, or to disk or
Rigid Disk Block
 from the Device Editor.

1.94 720c

The Analyze button starts up the device analyzer, based on the ↩
 device
information entered in the
Device Editor
. The Device Editor will keep
this button ghosted until enough data has been entered to make such a scan possible. When an analysis run is selected, the scanner will start up and

display any volumes that have been encountered. Internal device models will be constructed for any volume found during this scan. These can be saved to disk via the
Save Device
button, or to disk or
Rigid Disk Block
from
the Device Editor.

For analysis, a simple AmigaDOS pattern can be entered. This acts as a filter against volume names -- only volumes that match this pattern will be created. When no pattern is specified, all volumes found are generated.

1.95 720d

Click on this button to cancel the Device Editor/Analyzer operation and return to the input window. Since the device editor works in temporary storage, no changes are retained.

1.96 deveditmenu

The Device Editor has a single Project menu. When the Device Analyzer calls up the Device Editor, there are only simple
Help
and
Quit
menu items.
In Edit mode:

Menu Options:

Help

Load from RDB...

Save to RDB

Load from File...

Save to File...

Quit

1.97 devedithelp

The Help menu item displays help for either the
Device Editor
 proper
or the
Device Analyzer
, depending on the mode selected for the editor.

1.98 720e

The Quit item, like the close gadget, unconditionally quits DiskSalv. There is no difference between the two, though some users prefer one method over the other.

1.99 rdbinout

The Device Editor has functions to read or write a disk's rigid disk block.
The Rigid Disk Block is a standard for Amiga disk drives that allows partitioning information, among other things, to be stored on-disk, in a controller-independent fashion. When a hard disk device starts up, the partitions found in the RDB are examined and, usually, automatically mounted as AmigaDOS devices.

If something goes wrong with the RDB, the partition or partitions on the disk may be inaccessable by normal means. DiskSalv can help here. The

Device Analysis
 function can find any partitions physically present on a disk. These show up in the standard
Device List
. These can be saved back to the RDB via the
Save to RDB
 menu item. Alternately, the device editor can be loaded by selecting the
Load from RDB
 menu item.

1.100 7211

This function loads a device description from a disk's Rigid Disk Block into the device editor. This can be used to edit the sttings in the RDB, or simply to examine them. The name and unit number of the device must be entered in the device editor before this function can be selected.

1.101 7212

This function saves the device description in the Device Editor out to a disk's Rigid Disk Blocks. The Exec name, AmigaDOS name, unit number, and all numeric parameters must be entered before this can be done.

1.102 7214

The Save to File item brings up a standard file requester to allow the currently edited device to be written out to disk in DOSDrivers format. If no device is entered, this item is disabled. This is virtually the same as the Save Device button function, except that it works on the device description in the Device Editor rather than the currently selected input window device.

1.103 7215

This button brings up a standard file requester, to allow an ASCII device description file to be loaded as a virtual device into DiskSalv. Such a file must be in the DOSDrivers file format. A DOSDrivers file contains the description of a single device, using the same notation as originally defined for the system-wide MountList file. DOSDrivers files have been preferred on the Amiga, rather than the single MountList, since AmigaOS 2.00. Rather than use the Load from File menu item, a DOSDrivers file may simply be dropped into the Device Editor window. This is virtually the same as the Load Device button function, only it operates on the Device Editor only, it doesn'taffect the current input window device.

1.104 7304

The object of Device Analysis is to find any viable disk partitions on the given device and unit. As such, the data entered in the Device Editor is not for a specific partition, but for the search that will hopefully produce some partitions. The search can be limited by matching encountered volume names against a normal AmigaDOS pattern, and also limited by the range of sectors given for the analysis.

There are two analysis options, which differ only slightly. The Search option will stop analysis after the first volume matching the supplied pattern is found. A Search run will produce at most one new virtual device entry in the Device List. The Analyze option will find as many partitions matching the supplied pattern as possible, within the constraints of the supplied physical parameters.

The Search and Analyze buttons are disabled until enough device data is supplied. Device data doesn't have to come entirely by hand. The current device description is transferred on entry to the Device Editor, just as with a normal edit. DOSDrivers files work as usual, too. Icons dropped behave a bit differently. The device decription resulting will show the range of all AmigaDOS devices on that particular Exec device and unit. This won't necessarily cover the whole disk, but it can if the first and last partitions are known to AmigaDOS. Note that while the analysis routine can't go beyond the limits set, it can go beyond the end of the disk (device drivers return an error here). So it isn't absolutely necessary to figure the last block properly, though errors do slow things down.

The Device Analyzer produces a virtual device entry for each partition it finds. These will show up in the device list as soon as analysis is complete. The Save Device function can write this out to a DOSDrivers compatible MountList file. The Device Editor can be called up to modify anything that's found, or to save the device description out to a Rigid Disk Block descriptor.

1.105 paramfields

The Device Editor fields are similar to some of the parameters ↩
used in

DOSDrivers
and MountList files. DiskSalv needs only the parameters
that are called for in the Device Editor, while these aforementioned
AmigaDOS conventions support a much greater number of parameters.

Fields:

Device Name

DOS Name

Pattern

Unit

Surfaces

Sectors/Cylinder

Low Cylinder

High Cylinder

Bytes/Sector

Sectors/Block

Flags

Memory Type

1.106 7200

This is the name of the Exec device, such as trackdisk.device, scsi.device, etc. This can be typed directly into the string gadget, or it can be selected via a list of all devices, called up by clicking on the associated list gadget. An Exec device is necessary for all Device Editor functions. Naturally, not all devices in a system can be used here, only arbitrary disk-oriented devices can. There is no way to determine automatically whether a device follows this convention or not, but DiskSalv will report any failure to open the specified device.

1.107 7210

The is the name of the DOS device, such as DF0:, DH1:, etc. This field is present when the Device Editor is called up via the editor button. All devices must have a unique DOS device name specified, even if they’re just being created for use in DiskSalv.

1.108 7201

This is a standard AmigaDOS pattern string. This field is present when the
Device Editor is called up via the analysis button. This pattern is used as
a template for volumes enountered during the analysis run. If the Search
button is pressed, the scan quits as soon as one match is made. If the

Analyze
button is pressed, any volumes matching this pattern are added
to the device list. If no pattern is physically entered, the AmigaDOS
pattern #? (match anything) is used by default.

1.109 7202

This numeric field specifies the unit number associated with the Exec
device selected above. Most disk-oriented device drivers support more than
one unit. Typically, there are four floppy disk units (0..3), two IDE units
(0 and 1), and eight SCSI units (0..7) associated with their respective
device drivers. A unit entry is required.

1.110 7213

This numeric field specifies the number of surfaces physically supported on
a disk. Disk drives typically have several physicaly read/write heads, each
of which addresses a single disk surface. Floppy disks have two such heads,
one for the top surface, one for the bottom. Large hard disk drives may
have 15 or more active surfaces.

1.111 7206

This numeric field specifies the number of sectors physically located on a
disk's cylinder. For most devices this number is all but meaningless, since
devices are logically addressed. The proper cylinder size, if known, can
make some difference in speed, since buffering in cylinder-sized chunks can
increase efficiency. This parameter is more important for floppy disks,
since they're naturally buffered by the Amiga OS and can only be read or
written to in cylinder-sized chunks. For SCSI and other logically addressed
devices, its the aggregate cylinder, surface, and sectors/cylinder numbers
that are important taken together.

1.112 7203

This numeric field specifies the lowest logical cylinder (sometimes called
track) on the disk. Nearly every kind of disk and device driver starts the
disk at location zero. However, when creating a single device description,
it is the starting sector of the logical partition that's really the object

of concern. Analysis operations aren't concerned with starting or finishing at any specific point on a disk, but unless the approximate location of the missing partition is known, it is a good idea to start the analysis near the first cylinder of the disk.

DiskSalv device scans, and some other AmigaDOS device-oriented operations, express the start of the partition in terms of sector number. The relationship between cylinders and sectors is expressed as:

```
Sector = Cylinder * Sectors/Cyl. * Heads@
```

On physically addressed devices, like the floppy or the ancient ST-506, these numbers have significance to the device driver, but have never been a real concern to anything communicating to file systems or device drivers.

1.113 7204

This numeric field specifies the highest logical cylinder on the disk. Every disk has a physically defined last cylinder, but when creating a single device description, it is the last cylinder of the logical partition that's really the object of concern. Analysis operations aren't concerned with starting or finishing at any specific point on a disk. If the last cylinder on the disk isn't known, an arbitrary high value can be used. If the scanner runs off the end of the disk, DiskSalv will report disk errors. At this point, the operation can be stopped.

1.114 7207

This cycle gadget specifies the number of bytes per sector. All legal values are in the cycler. All file systems in AmigaDOS 1.3 and earlier used 512 bytes/sector, and that's still the most commonly used value today. Since the AmigaDOS 2.00 file system, larger values have been possible. Some versions of the file system had problems with very large blocks, but this is not a problem. When used in analysis runs, DiskSalv will attempt to reject blocks encountered that are formatted with a different byte/sector value.

1.115 7209

This cycle gadget specifies the number of sectors per block. This mechanism is another way to get larger blocks in a partition. It has the advantage over large bytes/sector setting in that every partition on the disk can easily have a different logical block size. All legal values are in the cycler. All file systems in AmigaDOS 2.1 and earlier used one Sector/Block, and that's still the most commonly used value today. Some early device descriptions didn't set this to one, so DiskSalv by default disables this field for older file systems (this can be overridden by setting the

BIGBLOCKS

command parameter).

Since the AmigaDOS 3.0 file system, larger values have been possible. The analysis routine attempts to reject blocks formatted under any size but the one selected here when scanning the disk. If a disk has been formatted with partitions of differing settings, several analysis passes may be necessary to find all partitions.

1.116 7205

This numeric field takes a startup code that is specific to the device driver in use. Most drivers don't do much with this field. A value of zero is the default, and recommended if there's no other value suggested by documentation or copying in any mounted device (such as the current device or one dropped in from Workbench).

1.117 7208

Many device drivers require a specific type of memory for their buffers, or at least perform better with a particular type. Floppies once required Chip RAM, though work properly with Fast memory in recent releases of the trackdisk.device. Most others work fine with system default memory. On a 32-bit Amiga, DMA-driven Zorro II based hard disk controllers work much better with DMA-24 memory, though most will use programmed I/O techniques to deal with other memory.

1.118 3202

The Disk Scanner is the primary progress indicator for actual ↩
DiskSalv
activity. Windows such as the
input
, Filter, or
Device Editor
are
used for setting up some kind of disk operation. Once a disk operation actually starts, DiskSalv brings up the Disk Scanner window. This is an informational display which provides a display of the current scanning function being run, a tally of various objects encountered, a progress indicator graph, and a list of major disk events.

The look of the scan window changes depending on several factors. The size of the window is based on the system font and the size of the screen being used. Several user options also control the look of this window.

Topic:

Displays

Button Options

1.119 scan.display

There are various displays within the Disk Scanner, as shown above ↩
. The
displays themselves are simple to explain. The Operation display indicates
the current type of scan being run. Most
Major Modes
are composed of
several scanner passes. The Device Scan display shows the current block,
plus a count of objects: files, directories, volumes, or errors, depending
on the
Major Mode
and scanner operation selected. The bar graph represents
the progress of the scan. In some operations, the progress indicated here
is an estimate. Finally, the Scanning Results display shows any major
significant results of the scan in progress, and can be instructed to pause
on any errors encountered.

Topic:

Operation Type

Device Scan Tally

Scanning Results

1.120 scan.operation

There are quite a few different scanning operations. There is no ↩
need to
understand any of these in order to use DiskSalv, but they're useful to
anyone interested in what's going on. The operation types are described
below.

Operations:

Checking Root

Cleaning

Copying

Directory Check

Extras

Filtering

FS Analysis

Hash Check

Link Check

List Trace

Loose Blocks

Paused

Purifying

Rehashing

Resolving

Salvaging

Scanning

Stopping...

1.121 scan.chkroot

In any
fix-in-place
mode, the disk's directories must be certified
as correct, or changed to be correct. The root block is a special case,
since no fix can occur if the root block cannot be corrected. The operation
used here is the same as in the Directory Check routine, but it is one of
the first operations run after a scan in all fix-in-place modes.

1.122 scan.cleaning

This operation indicates that the input disk is being cleaned of ↩
all
deleted files. This is the active part of the
Cleanup
mode. Once
a disk has been cleaned, none of the deleted objects remain, they are
completely wiped out and nothing can bring them back.

1.123 scan.copying

In this operation, objects from the input disk are simply copied to the
output disk. This is the active part of the Backup mode. The function is
similar to the Salvage operation, but since it runs on a presumably good
input volume, no special tricks are necessary to achieve the copy.

1.124 scan.chkdir

This operation is the actual
fix-in-place
function that certifies the
structure of every subdirectory on a disk. If there's a soft error in a directory block, it can rebuild this subdirectory as long as there's no physical problem with the disk.

1.125 scan.expanding

When the scanner is started by the
Stream Restore
function, this
operation is run. During such a run, a structured archive stream is rebuilt on the selected output device as an AmigaDOS directory.

1.126 scan.extras

Some modes have a small, mode-specific set of functions to run on the scanned data set. This operation name is a catch-all for these kinds of functions.

1.127 scan.filtering

When scanning takes place with a pattern selected, the pattern ↩
must be
applied to each object encountered. Some
patterns
, such as file name,
file note, date, size, or protection comparisons, are completely resolved during the scanning project, being local to the object. Other patterns can only be applied once a full scan is complete, such as full path comparisons. These are resolved here.

1.128 scan.analysis

This operation is run only when the
Best-Guess
pseudo file system
is selected. It chooses the most likely file system for the given disk, based on all of the objects encountered on that disk.

1.129 scan.chkhash

This operation, called during a
fix-in-place
operation, verifies the
integrity of every object as it appears on-disk. It does this by comparing the internal disk model to the contents of each directory as they appear on-disk. Each file encountered is checked completely by tracing out all data and list blocks that appear as components of the file. Anything that doesn't pass this test is eliminated from the active directory tree of the disk, but of course not physically erased from the disk itself.

1.130 scan.chklink

This is a
fix-in-place
operation that checks link objects. The hash
check routine will actually check that any links found are properly formed, and eliminate those that aren't. In this pass, interdependencies between links and the file they reference are handled. Any hard link pointing to an object that no longer exists will itself be removed. Symbolic links are processed for correctness, though the file system no longer supports them. No check is done on the object referenced by a symbolic link, since it can easily be on another volume.

1.131 scan.list

Files beyond a certain length (36864 bytes in FFS with 512 bytes/sector) use a linked list of list blocks to track additional file content blocks. DiskSalv tracks any such list blocks encountered during a scan in this phase, to locate possible partial files.

1.132 scan.loose

When scanning an Original File System disk, individual data blocks can be identified. In this phase, any data blocks enountered that were not assigned to a file are reconstructed as partial file nodes.

1.133 scan.paused

When the
pause
button is on, this is the operation displayed. The
previous operation resumes when pause is off. Other buttons, such as
stop
,

pause-on-error
, or the general kill gadget, still work while the
display and scan are paused.

1.134 scan.purifying

```
                This operation is the first
                fix-in-place
                 pass, called only for
                Repair
                mode.  It eliminates from the disk model any object that no longer ↩
                    exists
in the disk's directory tree, assuming that the tree is valid enough to
process in this way.  This prevents deleted files from being restored by
Repair.
```

1.135 scan.rehash

```
                Directory contents are presumed sorted in block sequence by the  ↩
                   fast file
system.  Once the main
                fix-in-place
                 operations have been run, this
function sorts the contents of all the directories on the disk.
```

1.136 scan.resolve

```
When a partial scan of the input disk is run, it's common for parent nodes
to be missed.  In this phase, any such unresolved parents are located.
```

1.137 scan.salvaging

```
                In this phase, the input disk's contents are being reconstructed  ↩
                   on the
output device selected on the
                output window
                .  This is the primary

                recover-by-copy
                 operation.
```

1.138 scan.scanning

```
                This operation is the preliminary scan of the input disk. There  ↩
                   are
different scanning funtions, depending on the
                Major Mode
                 selected,
but each builds a model of the input disk during this phase.
```

1.139 scan.stopping

This operation is displayed after the user presses the stop button. This indicates that DiskSalv has recognized the stop request, but must shut down some things before it can respond.

1.140 scan.tally

The Device Scan display is simply one kind of progress indicator. ↩
It shows a number of object counts, which vary by the Major Mode selected. Most of the time, the current block, a count of files, a count of directories, and a count of errors will be displayed. When the scanner is called up by the Device Editor, a running count of volumes replaces the error count. During a disk remake operation, a count of warnings and errors is kept.

In order to keep a scan running across the input disk as fast as is practical, the Device Scan display is normally updated only after a number of block counts. The frequency of update depends on the size of the input disk. Any error encountered will force an immediate update of the display.

No matter the optimizations, it's a simple fact that the time spent providing this display, as well as the Scanning Results display, does take some time away from scanning. When the Quick Scan option is selected from the input window's Settings menu, the scanner is opened as a much smaller window, with less frequent Device Scan updates. There is a progress indicator, but no Scanning Results display. In most cases, this has a noticable effect on scanning speed, especially when the system's CPU and hard disk are very fast.

1.141 scan.results

The Scanning Results display chronicles every major event that ↩
takes place during any kind of scan. This is the same information that is written to a log file, if such a file has been created on the input window. All the information can scroll by rather quickly, but the pause-on-error feature

button can be set to let the user see each error or warning as it scrolls past.

Each type of event known to the scanner is given a unique four character code. This is especially useful in log files, since it allows a search to be done for any event. However, it's also of general use, in that it completely identifies each event. The event codes include:

CHEK

FILE

ROOT

DATA

FLNK

SLNK

DSCH

FREE

UDIR

DELD

GOOD

WASH

DLNK

KILL

????

ERR!

LIST

1.142 event.chek

This code marks check failures. During a Check mode pass, possible errors are indicated but nothing is done about them. Check failures indicated on directories are usually repaired by a subsequent Repair mode run. Files, on the other hand, must be eliminated from the active disk tree when any component block shows a check failure.

1.143 event.data

In a scan of a disk formatted with the original file system, typed data blocks are identified by this code. Under the fast file system, data blocks are untyped.

1.144 event.dsch

Directory cache blocks are indicated by this code during a scan. These are only meaningful under the new directory caching file systems. Damaged directory cache blocks are usually not a problem, since DiskSalv can force the file system itself to rebuild them.

1.145 event.deld

This code is used during fix-in-place functions to indicate objects that have been judged as already deleted. In Repair mode, there is no desire to restore any objects that are already deleted. In Unformat mode, DiskSalv does attempt to bring back as much as possible, deleted or not, since it must be less trusting of the existing disk format.

1.146 event.dlnk

Hard directory links are indicated during a scan with this code, as long as they're considered valid. Like other objects, if the directory link must be removed from the active disk tree, it will be shown with the KILL code during the hash check phase of the fix-in-place routines. Links of all kinds are tallied with the file count for the Device Scan display.

1.147 event.err

Any kind of read error returned from the input device's driver is reported with this code. These are usually some kind of hard failure on the input disk.

1.148 event.file

This code is used to mark a normal file encountered during most scanning operations. This just indicates the file header block, which identifies the file by name, and indicates where the first group of file content blocks are. Other components to a non-empty file will be stored elsewhere on the disk. Due to the AmigaDOS disk format, it is possible for DiskSalv to find only part of a damaged file. It is left up to the user to determine whether a partial file is of any use in any given case.

1.149 event.flnk

Hard file links are indicated during a scan with this code. Like other objects, if the file link must be removed from the active disk tree, it will be shown with the KILL code during the hash check phase of the fix-in-place routines. Links of all kinds are tallied with the file count for the Device Scan display.

1.150 event.free

This code is displayed during scans of original file system disks to indicate a block that hasn't been used. There's no way to tell on a block-by-block basis that the block is unassigned. Since all OFS blocks are typed, though, any untyped block can be assumed as free.

1.151 event.good

This code is used during a fix-in-place operation when a disk object has been judged properly formed. Directories are judged during the Directory Check operation, while other objects are judged during the Hash Check operation.

1.152 event.kill

When an object within the input disk's directory tree has been judged
flawed and unrepairable, DiskSalv will remove it from the active directory tree, and display the object with the KILL code. This is necessary when a file component or link block is damaged physically or logically, or when a directory block is damaged physically. The
fix-in-place routines can
always reconstruct a directory that has been logically damaged, though this may cause the undeletion of some former files that still reside in such a directory.

1.153 event.list

This code identifies any file list blocks that are encountered during a
scan. An
AmigaDOS file format stores a fixed number of block pointers to
a file's content blocks within its header block. When the file is larger than this number of pointers can support, a list block (sometimes called an extension block) is created. Each extension block can access another set of file content blocks.

1.154 event.root

This code is displayed when a scan encounters a disk's root block. Each valid partition has a single root block, which is in the center of the disk. This block is much like a user directory block, though it stores some volume-specific data.

1.155 event.slnk

A symbolic link block is indicated by this code. The current file systems no longer support symbolic links, but they were partially supported in earlier releases (AmigaOS 2.x). DiskSalv knows how to handle a symbolic link based on the original specifications for it, and it will attempt to do so if it encounters such an object. The fix-in-place routines can verify that the on-disk structure of any symbolic link is correct, but since symbolic links can reference alternate physical devices, no existence check is done for the linked object.

1.156 event.udir

This is the code displayed for any subdirectory objects encountered during
a scan. DiskSalv can supply the missing data for any damaged directory in
any mode, though it can't fix a directory that's located on a physcially
bad block.

1.157 event.wash

During a
Cleanup
operation, any object that is permanently eliminated
is indicated by this code. Only objects that have been deleted are removed,
and only on a valid partition.

1.158 event.unkn

When a fast file system scan encounters an untyped block, it uses this
code. Since FFS data blocks are untyped, there is no way to determine if a
block is assigned or not from the block itself.

1.159 scan.buttons

The Disk Scanner window has three option buttons. These can be ↩
used to
control the progress of the scan, or simply stop it as quickly as possible.
Of course, there is also a standard close gadget on the window itself,
which will stop the scan and quit DiskSalv as quickly as possible.

Options:

Stop

Pause

Pause On Error

Ask On Error

1.160 3200

This button signals the scanner to stop the current operation as ↩
quickly as
possible. It will immediately cause the Operation display to indicate
Stopping... In many cases, this causes an immediate termination of the
scanning. The initial disk scan in any mode can be stopped immediately,
for instance. In other cases, the current processing must continue for
awhile in order to leave the input or output disk in a proper state. The

fix-in-place
routines, in particular, must completely finish processing once they start modifying the input disk. On output to another volume, files will not be broken up, so processing continues until the current file has been restored.

1.161 3201

The pause button simply pauses the scan where it is. This allows the user to view the Scanning Results display, check statistics, etc. The pause button is a toggle button; a single click pauses the scan, a second click resumes the scan. When running in pause-on-error mode, any error will pause the scan by setting the pause button. The pause button does not lock out input; all other gadgets on the scanner display work normally.

1.162 3204

The pause-on-error button controls an auto-pause feature of the scanner window. When on, any error or warning code found during a scan will cause the pause button to be automatically set. It can be toggled off to continue the scan just as if it were turned on by a user.

1.163 3205

The ask-on-error button controls and auto-inquire feature of the scanner window. When on, any error encountered during a fix-in-place scan that requires DiskSalv to modify the input disk will be prompted. The user may decide to perform the fix or ignore it. Clearly, if the error isn't corrected, DiskSalv won't fix the disk.

1.164 430b

The Output Window is brought up when DiskSalv has something that must be recovered by copying. A recover-by-copy operation is the primary function run by Salvage or Undelete modes, and a secondary function

```
run by any
                fix-in-place
                 mode that must eliminate files in order to
render the input device format valid.

  Topics:

                The Browser

                Path Setup

                Project Menu

                Settings Menu

                Salvage
                This window allows selection of files and output device for the  ↩
                   salvage
operation.
```

1.165 outputbrowser

```
                The Browser is where files are selected for recovery to another  ↩
                   device.
When DiskSalv enters the browser as the result of a
                recover-by-copy
                or
                backup
                 operation, no files are selected. The files present are
those that have been passed by any
                pattern
                 filtering set up by
the user.

The Browser consists of two custom list views. A view entitled "Select
Directory" is on the right hand side of the
                output window
                , while on the
left hand side is a similar view entitled "Select Files". The Directory
list contains a structured list of all the directories encountered, while
the File list contains a flat list of all files contained in the current
directory.  Each browser responds to an item selection, as well as the
selection of a number of extra function buttons along the right hand side
of the browser.

Each list supports the notion of current and selected. Some operations take
place on the single current item in each list, some operations take place
on all selected items in either or both lists. The final file recovery/copy
only includes those items that have been selected.  Browser operations
include:

    Directory   File

                Click on Item
```

Click on Item

Info

Info

Parent

Select

Select

Clear

Clear

Forget

Forget

1.166 4201

A mouse click on a directory
browser
item sets that directory as the
current directory. When a directory is current, its file contents are
displayed in the file
browser
. A directory is only indicated as selected
when all of its contents are selected.

1.167 4207

Click here for information on the current directory. This information
includes the directory's block number, date, and protection attributes.

1.168 4204

Click here to pick the parent of the current directory as the new current
directory. This item is ghosted if the current directory is the root
directory.

1.169 4205

Click here to select the current directory and all its children.

1.170 4202

Click here to return all selected items in the current directory and all its children to their unselected state.

1.171 4206

Click here to erase from working memory all currently selected items, including directories and files. This has no effect on what is on disk, it's provided simply as an aid to the user, to help in organizing a recovery or backup.

1.172 4214

A mouse click on a file browser item sets that file as the current file. Clicking on a file item also toggles selectionof the file item.

1.173 4216

Click here for information on the current file or link. This information includes the file's block number, size, date, and protection attributes.

1.174 4208

Click here to select all of the files in the current browser level. This works only at this level, not on any other branches of the parent directory.

1.175 4203

Click here to return all selected files in the current browser level to their unselected state. This works only at this level, not on any other branches of the parent directory.

1.176 4215

Click here to erase from working memory all currently selected files at this browser level. This has no effect on what is on disk, it's provided simply as an aid to the user, to help in organizing a recovery or backup.

1.177 outputpathsetup

```
                All disk recovery/backup data must be written out to an alternate ↩
                   AmigaDOS
device of some kind. Devices can be selected by name, optionally via an
AmigaDOS file requester. DiskSalv provides a special
                stream
                 format,
which stores a whole disk tree as a flat file,suitable for output to a file
or pipe.

  Topics:

                Output Path Gadget

                Output Path Requester

                Output Mode Select
```

1.178 4200

```
                Click on the Output Path gadget to specify the output device or ↩
                   path as a
string.
                File system
                 level recovery can be to any file system device
or directory, while
                streams
                 can go to files, pipes, or pipe-like devices
such as the SER: device. DiskSalv will create the directory or file if
necessary.
```

1.179 420a

```
                Click here to call up a standard file requester to select the ↩
                   output device
or path. This selection will be loaded into the
                Output Path Gadget
                 when
completed.
```

1.180 4213

```
                This cycle button allows selection of the output mode, which ↩
                   determines how
DiskSalv will treat the device entered in the
                Output Path Gadget
                . If
a
```

file system
output is indicated, DiskSalv will output a normal AmigaDOS file tree structure to the indicated device. Directories will be created as necessary, but output to a file or pipe will be displayed as an error. If a
stream
output is indicated, DiskSalv will format the recovery data as a flat byte stream, which can be output to a file on a valid file system device, as well as pipes and pipe-like devices such as PIPE:, SER:, or
TAPE:

1.181 outputproject

The
output window
has a project menu, which manages global help and navigation to other parts of the program.

Topics:

Help

New Device

Quit

1.182 outputhelp

This item simply calls up the main
output window
help page.

1.183 4211

Select this item to return to the
input window
. The results of the current scan are erased from memory, nothing additional is done to the input disk.

1.184 4212

Select this to quit the DiskSalv program.

1.185 outputsettings

This menu controls the settings of several simple
output window
parameters.

Topics:

Size Check

Warning Notes

Restore File Notes

Restore Protection

Restore Dates

Save Options

1.186 420b

Check this item to enable free space checking on the
output device
. Some
devices, like the RAM: device, always claim to be full. Output to such a
device will fail if this item is checked. This corresponds to the

NOSIZECHECK
command parameter.

1.187 420c

During a recovery, damage is sometimes detected to a recovered ↩
file. When
this item is checked, warning notes to this effect are written to the
FileNote field of the affected file or directory. This corresponds to the

NOWARNING
command paramater.

1.188 420d

When this item is checked, FileNotes are restored to files and ↩
directories.
When unchecked, they are ignored during recovery. This corresponds to the

NONOTES
command parameter.

1.189 420e

When this item is checked, file protection is restored to files and
directories. When unchecked, default protection is used during recovery.
This corresponds to the
NOPROTECT
command parameter.

1.190 420f

When this item is checked, date stamps are restored to files and directories. When unchecked, the current date is used during recovery. This corresponds to the
NODATES
command parameter.

1.191 4210

Pick this to save the selected options to the DiskSalv icon. The update is actually done when DiskSalv exits.

1.192 4209

Click here to start the disk salvage or backup. This button will stay ghosted until at least one item has been selected for recovery, and a valid output device of some kind has been entered.

1.193 appendix

Additional information is available on the following topics:

DiskSalv Support Files

Glossary of Terms

Command Parameters

DiskSalv Archival Format

Memory Requirements

AmigaDOS Disk Format

DOSDrivers Files

The DiskDoctor Story

1.194 supportfiles

There are a few files included with DiskSalv, which are designed to agument its normal operation. The files include:

DiskSalv.guide

DiskSalv looks for its online help file, DiskSalv.guide (this file, incidently), in its home directory. Depending on your AmigaGuide setup, it may be found elsewhere as well.

DiskSalv.catalog

DiskSalv looks for translation catalogs in the normal LOCALE: directory. They should be named DiskSalv.catalog. DiskSalv2 catalog files will produce strange results; they can be loaded, but are not recommended.

DiskSalv 3 has not yet been translated, so there is no current DiskSalv.catalog.

DiskSalv.pattern

DiskSalv looks for a default Pattern file, named DiskSalv.pattern, in its home directory. Any number of other pattern files may be manually loaded, via the Pattern Load button.

1.195 glossary

Important terms in the understanding of DiskSalv include:

Device, DOS

Device, Exec

Disk

File System

Hard Erorr

Partition

Pattern, AmigaDOS

Pattern, DiskSalv

Rigid Disk Block

```
Root Block

Soft Error

Streams

TAPE:

Tripos

Volume
```

1.196 glossary.adospattern

```
The AmigaDOS pattern matching language defines a flexible syntax for
defining regular expressions.  The language include:

    normal
  Any normal character matches itself.

    'special
  The quote suppresses the special action of a special character.

    ?
  Matches any single character.

    %
  Matches an empty string

    [normal low-normal high]
  Matches a range of characters.

    ~expression
  Specifies a pattern matching anything expression doesn't match.

    #expression
  Specifies a pattern matching zero or more of expression.

    (expression)
  Makes a sub-pattern of expression.

    expression 1|expression 2
  Matches expression 1 or expression 2
```

1.197 glossary.adosdate

```
AmigaDOS specifies file/directory dates using the following syntax:

  day_number-month_abbrev-year_number

Examples include:
```

```
  23-May-61 May 23rd, 1961
  24-Apr-90 April 24th, 1990
  17-Jul-91 July 17th, 1991
  12-Mar-94 March 12th, 1994

Month abbrevs are: Jan, Feb, Mar, Apr, Jun, Jul, Aug, Sep, Oct, Nov, Dec
```

1.198 glossary.dspattern

```
                DiskSalv extends the AmigaDOS regular expression language into a  ↩
                   flexible
pattern matching language. DiskSalv complex patterns can match against
directories, files, dates, file sizes, and filenotes. They can be used to
include or exclude the matched item, and to stop the DiskSalv scanner on a
match.  See
                Patterns
                 for more information.
```

1.199 glossary.dosdevice

```
                An AmigaDOS "device" is a often named physical disk or disk device ↩
                   .  If a
disk is fixed, the AmigaDOS device,
                partition
                , and physical
                volume
                 all
reference the same thing.  If the disk is removable, the AmigaDOS device
name references the physical drive mechanism, while the physical
                volume
                references a particular disk, independently of the particular  ↩
                   drive
mechanism.

A DOS device, which is basically just a file system server, is concerned
with high-level objects and complex commands.  DOS devices know about
files, directories, and links, and provide functions to open, seek, read,
write, and close files, access file support data, etc.

See also
                AmigaDOS File System
                ,
                Exec Device
```

1.200 glossary.execdevice

```
                An Exec device is the low-level device driver used by DiskSalv to  ↩
                   access an
input disk.  DiskSalv can only repair or recover from
                AmigaDOS devices
```

that are built on Exec devices. For example, the RAM: disk is a ↩
DOS
device, but it has no underlying Exec device. The DF0: floppy, however, relies on the "trackdisk.device" Exec-level device to handle low-level communications to the floppy hardware.

Exec-level devices are concerned with low-level objects and very simple commands. Such devices are treated as arrays of fixed-sized blocks, and commands are supplied to read, write, or format such blocks. Other commands can control a drive's motor and various other functions of the physical disk drive.

See also
AmigaDOS Device

1.201 glossary.filesystem

A file system is a program that fields high-level commands about ↩
AmigaDOS
structures: Files, Links, Directories, and the various other accounting data on a disk. Each AmigaDOS device is represented as a file system task. So, for instance, if I have a hard disk named "DH0:" as seen from AmigaDOS, there will be a task running, say, the L:FastFileSystem program, and that task's name will be DH0.

There are several types of file systems in the Amiga Disk Operating System. Some, like the RAM: device, are of little concern to DiskSalv. DiskSalv is primarily concerned with standard disk-based file systems. There are currently six variations of the AmigaDOS file system. A file system can use the original, international, or directory-caching storage mechanism, and its block organization can be "original" or "fast". The original block structure is a bit more robust (e.g. easier to repair or recover), the fast structure is faster.

See also
AmigaDOS Device

1.202 glossary.disk

While the term "disk" can be a bit vague, it's generally used as a ↩
synonym
for the physical
partition
or device under consideration.

1.203 glossary.harderror

A "hard error" is a physical defect of some kind on the input disk ↩
. Hard
errors are usually the result of some physical failure on the disk, and can

not be repaired by DiskSalv. It's usually necessary to reformat any disk that has hard errors, electing to map such blocks as "bad".

See also
Soft Error

1.204 glossary.partition

A disk partition is the most general physical instance of an AmigaDOS disk. While a disk can be set up as a single entity, as with most floppies, it's usually more useful to divide large disks into several subsections, or partitions. Each partition will have an AmigaDOS device name, an AmigaDOS volume name, and any number of logical assigns depending on the system setup.

1.205 glossary.rdb

The Rigid Disk Block, or RDB, is a convention for storage of useful boot-time AmigaDOS data, in a controller-independent format. By convention, the first cylinder or two of a hard disk (or other partitionable AmigaDOS device) is reserved for the RDB. A device driver can simply start reading this track to find data on the user-defined partitions, file systems, bad block mappings, and other data for the disk. Since this was standardized reasonably early in the evolution of Amiga hard disk controllers, nearly every such controller supports the convention. So AmigaDOS hard disks can be freely mixed and matched between system and controller, without need to configure anything on the host system.

1.206 glossary.rootblock

All tree-oriented disk structures support the concept of a "root", or top-level directory. In AmigaDOS file systems, there is a root block on each partition. This block is located in the center of the disk.

1.207 glossary.softerror

A "soft error" is a logical defect of some kind on the input disk. ↩
Soft
errors are usually created when the system fails during a disk write. This can be indirectly due to a physical act: premature disk removal, power failure, disk controller failure, keyboard reset, etc. This can also be due to a failure in software, (e.g. a program crash). While the Amiga Operating System can trap many types of software failure, it does not support memory protection and therefore cannot offer perfect protection. An errant program can cause the AmigaDOS
file system
or a disk's device driver to fail,

resulting in a soft error. A soft error may be repaired by DiskSalv, depending on its location on the disk. Soft errors rarely result in any need for reformatting of the input disk.

See also
Hard Error

1.208 glossary.streams

DiskSalv has the capability to format any
recover-by-copy
or
backup
data as a flat byte stream. This byte stream preserves the directory tree
structure as found on the input disk by creating typed blocks within the stream file for each element in the original disk tree. Such data can be sent to any standard AmigaDOS pipe or pipe-like device, such as PIPE:, SER:, or TAPE:.

While DiskSalv does not provide any sort of data compression, this mechanism can be used to compress this output. For example, let's assume the user has a hard disk, DH0:, and a problem disk DH1:, which is too severely damaged to fix in place. The user runs
Salvage
on the disk,
and gets everything listed. But it won't quite fit on DH0:. Rather than panic, or resort to copying out onto floppies, the user can run everything through a compression tool, such as compress. Any compression program that will accept input from a pipe (eg, the standard input) will work.

On the
output window
, one would enter a named pipe, say PIPE:recover
for example, into the
Output Path
gadget, and specify a stream output
to the
Output Mode
gadget. Now, in a separate shell, one would enter something like:

```
1> compress <PIPE:recover >DH0:dh1_stream
```

Once the recovery is complete, the file dh1_stream should contain the compressed contents recovered from the damaged DH1: device. Reformat the DH1: device, then restart DiskSalv. In a separate shell, enter something like:

```
1> uncompress <DH0:dh1_stream >PIPE:recover
```

Start up DiskSalv, select the
Restore Stream
button. Enter PIPE:recover
as the input device, the newly reformatted DH1: as the output device.

DiskSalv will now restore the complete DH1: contents. After checking that
everything's correct, the dh1_stream file can be deleted.

See also
                Stream Format

1.209 glossary.tapedevice

DiskSalv can support most TAPE: devices, but doesn't come with one. A TAPE:
device is a standard AmigaDOS file handler interfaced to some kind of tape
backup device. If you buy a backup unit specifically for use with the
Amiga, it should come with such a device, though it may have a different
name. With older systems, you add the vendor's specified MountList example
to your Devs:MountList file, editing it if necessary to specify the device
(most tape drives use a SCSI controller, like Commodore's "scsi.device")
and unit number, as approproiate.

If your tape drive didn't come with a tape handler, you may be able to find
one that's freely redistributable. The one I recommend is BTNtape, by Bob
Rethemeyer. While this was originally a very basic tape handler, as the
"Better Than Nothing" monniker would imply, today it's fairly complete.
Best of all, it has support for a number of different tape drives. Unlike
hard drives, SCSI tape drives are a bit quirky with respect to their
command sets, so customization to the specific tape drive is often
required. BTNtape is on the Fred Fish CD-ROMs, older versions are in the
Fish floppy collection.

1.210 glossary.tripos

The Tripos Operating System is an OS that attained a reasonable level of
popularity in the UK. Due to time constraints during the development of
the Amiga Operating System, Metacomco Ltd. was contracted to adapt the DOS
subsystem of Tripos to the Amiga OS.

1.211 glossary.volume

A "Volume" in AmigaDOS is any logical device. All volume names end in ":",
and come in two flavors. Physical volumes are names associated with
particular disks or disk partitions. If a disk is fixed, its device name
and physical volume name reference the same item. With removable disks,
the device name references the physical drive (such as DF0:), while a
physical volume name references a particular disk (such as "Workbench").
Logical volume names are created with the "Assign" command in AmigaDOS, and
can reference an AmigaDOS device, physical volume, or subdirectory.

1.212 cli

Command parameters are options specified to DiskSalv as command- ↩
line or
ToolTypes keywords. Different setups can be created via multiple Project
icons.

ASKONERROR/S

BIGBLOCKS/T

DEFAULTFS/K

DISKCACHE/N

FILESYSTEM/K

FORCEGUIDE/S

FONT/K

FROM/K

INTERACTIVE/S

KEEPDOS/S

KILLDOS/S

LOADDEV/K

LOWMEM/S

MAKELINKS/S

MEMCHUNK/N

MODE/K

NOARCHIVE/S

NODATES/S

NODEEPSCAN/S

NOGUIDE/S

NONOTES/S

NOPROTECT/S

NOSIZECHECK/S

NOWARNING/S

PATHMAX/N

PAUSEONERROR/S

```
PUBSCREEN/K

QUICKSCAN/S

REJECTION/N

RETRY/N

SMALLWINDOW/S

TAGCHAR/K

TO/K
```

1.213 cp.askonerror

```
This command parameter presets the
Ask on Error
 mode of the scanner,
causing DiskSalv to automatically prompt the user before making any
modifications to the input disk, during any
fix-in-place
operation.
```

1.214 cp.bigblocks

```
This lets the user specifiy whether support for multiple sectors per block
is enabled.  On pre-V39 systems, occasionally DOS devices claimed for some
reason to have more than one sector per block.  Since real support of this
didn't appear until the V40 FileSystem, which apparently needs V39 to run,
multiple sector per block support is off by default in on a pre-V39 system,
on by default for V39 systems and above.
```

1.215 cp.defaultfs

```
This allows the user to specify the fallback file system to use on ↩
    the
input device.  Normally, DiskSalv will determine a partition's file system
from its root block, but if that root block is damaged, DiskSalv needs this
fallback. Normally this is the
Best-Guess
 pseudo file system, but
it can be changed here.  Please see
File System Selection
 for more information.
```

1.216 cp.diskcache

This specifies the number of blocks to be used for the disk cache (which is actually a pre-fetch buffer). The default size is 8 blocks, and can be set between 0 (no cache) and 255 blocks.

1.217 cp.filesystem

This allows the user to specify the file system to use on the input device.

Normally, DiskSalv will determine a partition's file system from its root block. If for some reason that information is unavailable or wrong, this allows the file system type to be specified by force. The supported file systems include:

```
OFS   Original File System.
FFS   Fast File System.
OFS Intl. OFS with ISO 8-bit character support.
FFS Intl. FFS with ISO 8-bit character support.
DC-OFS    OFS with directory caching.
DC-FFS    FFS with directory caching.
```

Best-Guess
This pseudo file system selects the best match.

DiskSalv can detect, but not actually process, several other file system types, including some MS-DOS types. Please see File System Selection for more information.

1.218 cp.forceguide

This switch is now no longer necessary. Originally this was designed to suppress the loading of the DiskSalv guide file for versions of the amigaguide.library before V39. This latest version of DiskSalv incorporates work-arounds to the bugs in these older guide libraries that allows safe operation. Both hot links from DiskSalv or direct loading by the AmigaGuide program work fine.

1.219 cp.font

This specifies a font and point size to use, rather than the system default font. DiskSalv will use this font only if it allows the DiskSalv window to size properly. If not, the screen and then system defaults will be tried, with a final drop back to topaz 8, which always works. On the CLI, this should be specified as font/point or font,point. As a tool type, font point also works.

1.220 cp.from

This keyword specifies an input device, the device that will be operated on. This must be a true AmigaDOS device, not a subdirectory or assignment, and it must be based on one of the standard file systems with underlying device driver.

1.221 cp.interactive

This forces DiskSalv into interactive operation (eg, it waits for user input). When run from the CLI, DiskSalv will by default do as much as it can non-interactively before going into interactive mode.

1.222 cp.keepdos

The FileSystem (eg, AmigaDOS) is usually locked out from the input device during scan and recovery operations. This option will prevent such a lockout. Fix-In-Place operations are not affected by this, as they absolutely requie a FileSystem lockout since they are modifying the input disk. Note that allowing any writes to the input disk during a DiskSalv operation will almost certainly cause DiskSalv to malfunction.

1.223 cp.killdos

This option forces DiskSalv to do its job without using any device directed DOS functions or file system packets. Normally, DiskSalv uses a handfull of DOS library functions on input disks. This is designed to make DiskSalv safe for use on disks that crash AmigaDOS or the particular file system in use. This is a somewhat dangerous option, since it prevents DiskSalv from locking the file system out of a device (since that requires a file system packet). It works best on devices that can't be given to AmigaDOS because they're too damaged. Note that DOS is automatically avoided on unmounted devices.

1.224 cp.loaddev

This argument takes the name of a
DOSDrivers
compatible device
description file, which it will load and set as the current input device.

1.225 cp.lowmem

This specifies low-memory mode. If DiskSalv runs out of memory on a system in normal mode, it may be successful in low-memory mode. This automatically causes the chunky allocator, file path buffer, and disk cache to go to minimum sizes. It cuts out a number of other internal things that generally just affect performance rather than success.

1.226 cp.makelinks

This causes DiskSalv to actually create links on the output volume. Normally it instead creates a script file which will create the links.

1.227 cp.memchunk

This specifies the memory chunk size for DiskSalv's chunky allocator to use. By default, this is 4K, and can be set between 1K and 128K.

1.228 cp.mode

There are several kinds of functions that DiskSalv will run. The ↩
modes
include
Salvage
,
Undelete
,
Repair
,
Unformat
,

Check
,
Backup
, and
Cleanup
. When running with a
localization, either the built-in or localized mode namesmay be supplied as arguments.

1.229 cp.noarchive

When in backup mode, DiskSalv normally sets the archival bit on any file that it backs up. This switch will prevent archive bits from being set.

1.230 cp.nodates

This option will inhibit restoration of the original file date in
Recover-by-Copy operations.

1.231 cp.nodeepscan

This inhibits extra low-level processing from being done floppy disks. Such
processing can recover data not normally accessable through the
trackdisk.device, but it causes extra memory to be used.

1.232 cp.noguide

This inhibits DiskSalv's opening of the AmigaGuide.library. A severe bug in
some early (V34) versions of the AmigaGuide library cause undefined
behavior, including drastic system crashes, if the AmigaGuide library
attempts to open a guide file but fails. This prevents DiskSalv from making
that attempt.

1.233 cp.nonotes

This option will inhibit restoration of the original FileNote in
Recover-by-Copy operations (though warning notes will override original
notes).

1.234 cp.noprotect

This option will inhibit restoration of the original protection codes in
Recover-by-Copy operations.

1.235 cp.nosizecheck

This inhibits automatic size checking of the output volume. Normally,
DiskSalv watches the size of the output volume to have an idea ahead of
time that a volume will fill up. Some devices, such as RAM:, are
dynamically sized and always indicate full when asked, so this parameter is
mainly intended for such devices (DiskSalv actually invokes this
automatically for RAM:, but it would have to be specified manually for
other such devices).

1.236 cp.nowarning

DiskSalv will normally attach a warning or error message, as a FileNote, to any file it restores via a Recover-by-Copy operation that it considers suspect or bad. This option will inhibit such action.

1.237 cp.pathmax

This specifies the maximum length of a file path. The default value is 512 bytes, and can be set between 256 bytes and 4K.

1.238 cp.pauseonerror

This command parameter causes all scan runs to set Pause on Error mode by default.

1.239 cp.pubscreen

This specifies the public screen, by name, for DiskSalv to start up on. If none is specified, DiskSalv will start up on the Workbench screen (eg, default public screen).

1.240 cp.quickscan

This specifies a faster disk scanning mode. The speed of a disk scan is improved by cutting down on the visual display. DiskSalv will still show a "gas-guage" indicator, but it won't list objects as they are encountered.

1.241 cp.rejection

This specifies a filter strength, between 1 and 10, that influences DiskSalv's assessment of whether or not a disk block matches a specific block type. This is generally left at the default, 6. In some cases, adjusting this may improve the performance of the scanner, depending on the disk problems at hand. It can also make the scanner perform worse.

Technically speaking, this supplies a normalized adjustment to the decision threshold in the fuzzy disk block matching routines within DiskSalv. Lower values allow less qualified blocks to be accepted, while the full scale 10 setting causes only pefect blocks to be accepted. The main problem with very low values is that block typing can suffer -- a directory may be seen as file, or vice-versa.

1.242 cp.retry

This argument changes the number of retries a device drive will run on a read failure. USE THIS OPTION WITH CAUTION! This function uses the convention of the trackdisk.device for retry count, which isn't guaranteed to be supported by other device drivers. In general, the default retry count is what you want to use. When a device has a large number of errors, it can be processed much faster by setting this parameter as low as zero, if the selected device supports this convention. If not, there's a chance it will cause some unknown problem, so it's best to use only as a last resort.

1.243 cp.skipdevs

This parameter allows the user to specify a list of devices that will be ignored by DiskSalv. For example, to ignore two network devices, "UNIX:" and "VAX:", let's say, the parameter is entered as:

```
SKIPDEVS UNIX:|VAX:
```

Any number of devices may be entered. The primary use of this is to restrict the DiskSalv device list from using invalid devices. DiskSalv can normally reject such devices anyway, but in some cases, it may not reject it correctly, or it may access invalid memory. This is primarily due to the fact that AmigaDOS doesn't provide a truely reliable way to find standard file system based devices.

1.244 cp.smallwindow

This causes DiskSalv to build a minimal scanning display window, even on large screens.

1.245 cp.tagchar

This is maintained only for command-line compatibility with DiskSalv 2. In DiskSalv 2, a character could be specified to indicate objects that are tagged in the file browser. This was done because the standard list gadget in AmigaOS 2.x could not do any kind of highlighting. DiskSalv 3 uses a custom list gadget for all its lists, and can therefore highlight selections in AmigaOS 2.x and 3.x.

1.246 cp.to

This keyword specifies an output device, the device that will receive any recover-by-copy files that DiskSalv finds. This may be any AmigaDOS device, volume, or subdirectory. If a non-existant subdirectory is specified, one will be generated.

1.247 streamformat

The DiskSalv Stream concept was designed to provide a reliable backup/archival mechanism for data processed by DiskSalv. This format retains everything DiskSalv can do with recovery or backup, but puts it in a more manageable form, which can be processed by compression utilities or directed to tape.

The DiskSalv stream format is based on tagged 512 byte blocks, regardless of the block size of the input disk. Each block starts with its type identifier, which always a 4-byte ASCII code:

ROOT
  The archive root/start

UDIR
  User directory header

FILE
  Normal file header

DATA
  Data belonging to a file

DLNK
  Directory link header

FLNK
  File link header

SLNK
  Symbolic link header

ERRS
  An error block

ENDA
  The archive's end

1.248 sf.root

This is the ROOT structure. All stream blocks are padded to 512 bytes.

```
LONG   'ROOT'    ; Block type.
LONG   pad[2]    ; Unused.
LONG   parent    ; The id number for the parent block.
LONG   id    ; The id number for the object.
LONG   checksum  ; A checksum code goes here.
char   name[32]  ; The archive name
DateStamp  created   ; The volume's creation date.
DateStamp  modified  ; The volume's modification date.
DateStamp  date    ; The archive's creation date.
LONG   barcount  ; A count of file objects
```

1.249 sf.udir

This is the UDIR structure. All stream blocks are padded to 512 bytes.

```
   LONG   'UDIR'    ; Block type.
   LONG   pad[2]    ; Unused.
   LONG   parent    ; The id number for the parent block.
   LONG   id    ; The id number for the object.
   LONG   checksum  ; A checksum code goes here.
   char   filename[32]  ; The File name.
   LONG   protect   ; The Protection field.
   DateStamp  date    ; The File date.
   char   filenote[92]  ; The FileNote.
```

1.250 sf.file

This is the FILE structure. All stream blocks are padded to 512 bytes.

```
   LONG   'FILE'    ; Block type.
   LONG   size    ; File size in bytes.
   LONG   count   ; File size in blocks.
   LONG   parent    ; The id number for the parent block.
   LONG   id    ; The id number for the object.
   LONG   checksum  ; A checksum code goes here.
   char   filename[32]  ; The File name.
   LONG   protect   ; The Protection field.
   DateStamp  date    ; The File date.
   char   filenote[92]  ; The FileNote.
```

1.251 sf.data

This is the DATA structure. All stream blocks are padded to 512 bytes.

```
   LONG   'DATA'    ; Block type.
   LONG   size    ; Byte size of the data block.
   LONG   count   ; Block sequence in file.
   LONG   parent    ; The id number for the parent block.
   LONG   id    ; The id number for the object.
   LONG   checksum  ; A checksum code goes here.
   LONG   data[]    ; The of the structure stores data.
```

1.252 sf.dlnk

This is the DLNK structure. All stream blocks are padded to 512 bytes.

```
   LONG   'DLNK'    ; Block type.
   LONG   pad[2]    ; Unused.
   LONG   parent    ; The id number for the parent block.
   LONG   id    ; The id number for the object.
```

```
   LONG   checksum  ; A checksum code goes here.
   char   filename[32]  ; The File name.
   LONG   protect   ; The Protection field.
   DateStamp  date    ; The File date.
   char   filenote[92]  ; The FileNote.
   LONG   link    ; File/Directory ID to link to.
   LONG   chain   ; Link chain, if any
```

1.253 sf.flnk

This is the FLNK structure. All stream blocks are padded to 512 bytes.

```
   LONG   'FLNK'    ; Block type.
   LONG   pad[2]    ; Unused.
   LONG   parent    ; The id number for the parent block.
   LONG   id    ; The id number for the object.
   LONG   checksum  ; A checksum code goes here.
   char   filename[32]  ; The File name.
   LONG   protect   ; The Protection field.
   DateStamp  date    ; The File date.
   char   filenote[92]  ; The FileNote.
   LONG   link    ; File/Directory ID to link to.
   LONG   chain   ; Link chain, if any
```

1.254 sf.slnk

This is the SLNK structure. All stream blocks are padded to 512 bytes.

```
   LONG   'SLNK'    ; Block type.
   LONG   pad[2]    ; Unused.
   LONG   parent    ; The id number for the parent block.
   LONG   id    ; The id number for the object.
   LONG   checksum  ; A checksum code goes here.
   char   filename[32]  ; The File name.
   LONG   protect   ; The Protection field.
   DateStamp  date    ; The File date.
   char   filenote[92]  ; The FileNote.
   char   link[]    ; The link name goes here.
```

1.255 sf.errs

This is the ERRS structure. All stream blocks are padded to 512 bytes.

```
   LONG   'ERRS'    ; Block type.
   LONG   junk[]    ; The rest is undefined.
```

1.256 sf.enda

This is the ENDA structure. All stream blocks are padded to 512 bytes.

```
   LONG   'ENDA'    ; Block type.
   LONG   pad[2]    ; Unused.
   LONG   parent    ; The id number for the parent block.
   LONG   id    ; The id number for the object.
   LONG   checksum  ; A checksum code goes here.
   LONG   pad   ; Unused
   LONG   filecount ; Number of files in archive.
   LONG   dircount  ; Number of directories in archive.
   LONG   linkcount ; Number of links in archive.
   LONG   errcount  ; Number of recorded errors
   LONG   objectcount ; Total count of objects.
   DateStamp  date    ; Date code for recovery set.
   LONG   pad[2]    ; Unused
   LONG   dwalloc   ; Total buffer size
   LONG   pad   ; Unused
```

1.257 memoryrequirements

DiskSalv attempts to conserve on memory, but there are definite
requirements that are fixed. These can be simply summarized:

```
  Item    Memory used

  DISK OVERHEAD   2 bits per block

  FILE      8 bytes
  LINK      24 bytes plus file name length
  DIRECTORY, FFS    24 bytes plus file name length
  DIRECTORY, DC-FFS 28 bytes plus file name length
```

1.258 amigadosformat

SECTION NOT COMPLETE

1.259 dosdrivers

DiskSalv can read and write Amiga DOSDrivers files. These files are
standard device description files, used by the AmigaDOS Mount command. The
DOSDrivers form of the mount file was originated in AmigaOS 2.1, as a
slight variation of the MountList form. In this file format, the AmigaDOS
device name is derived from the file name, and the file contains a single
device description.

The

Save Device
button on the
input window
and the
Save to File...
can save the current device to a DOSDrivers format file. This is
necessarily a complete file; some
parameters
may be determined
by the device driver type.

Similarly, a DOSDrivers file may be loaded via the
Load Device
button
on the
input window
or the
Load from File...
menu item on the

Device Editor
window. It may also be loaded by dropping it by
icon directly into either window. The
parameters
not of interest
to DiskSalv are ignored.

1.260 dosdrivers.params

The following DOSDrivers parameters are of interest to DiskSalv. These are used by DiskSalv to build internal device descriptions, and they're written out when DiskSalv creates a DOSDrivers file.

Parameter	Explanation
Name	Taken from the file name
BlocksPerTrack	Size of a track, in blocks
BlockSize	Longword size of a block
BufMemType	Type of memory, 0 for anything, 3 for chip
Device	Device driver name, like "scsi.device"
DosType	32-bit DOS type identifier
Flags	Flags for the OpenDevice() call
HighCyl	Highest cylinder on the partition
LowCyl	Lowest cylinder on the partition
Reserved	Number of blocks reserved on a partition
SectorsPerBlock	Number of sectors in a block
SectorsPerTrack	Size of a track, in sectors
SectorSize	Longword size of a sector
Surfaces	Number of active disk surfaces, or heads
Unit	Device driver's unit number

Note that older file systems don't support multiple sectors per block, and don't understand sector-based parameters. DiskSalv saves in terms of SectorsPerTrack and SectorSize if the SectorsPerBlock parameter is other

than 1. Otherwise, it uses the better supported BlocksPerTrack and
BlockSize parameters.

1.261 diskdoctor

DiskSalv's First Competition

When the Amiga was first introduced, much was made of the robustness of its

file system
. In the process of working my first Amiga software project, I ran into a disk error. Unfortunately, no tool was available at the time that would do anything about this error, so I started writing DiskSalv.

Shortly thereafter, the DiskDoctor program was introduced. Unlike my original DiskSalv, this program claimed to fix disks in-place rather by copying out to another volume. With floppy disks, the recover by copy mechanism wasn't much of a problem, but once hard disks became popular, the fix-in-place solution was the solution of choice.

Unfortunately, DiskDoctor was flawed. While it could repair a disk, it could also cause damage to a disk's structure. It did improve over the years, but was never considered to be reliable. Eventually it was removed from the Workbench entirely.

The Demise of the DiskDoctor

The story I was told goes something like this. The software folks were not quite sure whether DiskDoctor should be dumped or improved, so they decided to leave it up to DiskDoctor itself. They put the DiskDoctor sourced on an old floppy, then ran DiskDoctor on it. As often happened, DiskDoctor damaged this undamaged disk. So, while it's often said that DiskDoctor was "sued for malpractice", it's more correct to state that DiskDoctor committed suicide.