computers are bad

https://computer.rip - me@computer.rip - fax: +1 (505) 926-5492

2024-05-25 grc spinrite

I feel like I used to spend an inordinate amount of time dealing with suspect hard drives. I mean, like, back in high school. These days I almost never do, or on the occasion that I have storage trouble, it's a drive that has completely stopped responding at all and there's little to do besides replacing it. One time I had two NVMe drives in two different machines do this to me the same week. Bad luck or quantum phenomenon, who knows.

What accounts for the paucity of "HDD recovery" in my adult years? Well, for one, no doubt HDD technology has improved over time and modern drives are simply more reliable. The well-aged HDDs I have running without trouble in multiple machines right now support this theory. But probably a bigger factor is my buying habits: back in high school I was probably getting most of the HDDs I used second-hand from the Free Geek thrift store. They were coming pre-populated with problems for my convenience.

Besides, the whole storage industry has changed. What's probably more surprising about my situation is how many "spinning rust" HDDs I still own. Conventional magnetic storage only really makes sense in volume. These days I would call an 8TB HDD a small one. The drives that get physical abuse, say in laptops, are all solid state. And solid state drives... while there is no doubt performance degradation over their lifetimes, failure modes tend to be all-or-nothing.

I was thinking about all of this as I ruminated on one of the "holy grail" tools of the late '00s: SpinRite, by Gibson Research Corporation.

The notion that HDDs aren't losing data like they used to is supported by the dearth of data recovery tools on the modern shareware market. Well, maybe that's more symptomatic of the complete hollowing out of the independent software industry by the interests of capitalism, but let's try to dwell on the positive. Some SEO-spam blog post titled "Best data recovery software of 2024" still offers some classic software names like "UnDeleteMyFiles Pro," but some items on the list are just backup tools, and options like Piriform Recuva and the open-source PhotoRec still rank prominently... as they did when I was in high school and my ongoing affection for Linkin Park was less embarrassing [1].

Back in The Day, freeware, shareware, and commercial (payware?) data recovery software proliferated. It was advertised in the back of magazines, the sidebar banner ads of websites, and even appeared in the electronics department of Fred Meyer's. You also saw a lot of advertisements for services that could perform more intensive methods, like swapping an HDD's controller for one from another unit of the same model. These are all still around today, just a whole lot less prominent. Have you ever seen an Instagram ad for UnDeleteMyFiles Pro?

First, we should talk a bit about the idea of data recovery in general. There are essentially two distinct fields that we might call "data recovery": consumers or business users trying to recover their Important Files (say, accounting spreadsheets) from damaged or failed devices, and forensic analysts trying to recover Important Files (say, the other accounting spreadsheets) that have been deleted.

There is naturally some overlap between these two ventures. Consumers sometimes accidentally delete their Important Files and want them back. Suspects sometimes intentionally damage storage devices to complicate forensics. But the two different fields use rather different techniques.

Let's start by examining forensics, both to set up contrast to consumer data recovery and because I know a lot more about it. One of the quintessential techniques of file system forensics is "file carving." A file carving tool examines an arbitrary sequence of bytes (say, from a disk image) and looks for the telltale signs of known file formats. For example, most common file formats have a fixed prefix of some kind. ZIP files start with 0x504B0304, the beginning of which is the ASCII "PK" for Phil Katz who designed the format. Some formats also have a fixed trailer, but many more have structure that can be used to infer the location of the end of the file. For example, in ZIP files the main header structure, the "central directory," is actually a trailer found at the end of the file.

If you can find the beginning and end of a file, and it's stored sequentially, you've now got the whole file. When the file is fragmented in the byte stream (commonly the case with disk images), the problem is a little tougher, but still you can find a lot of value. A surprising number of files are stored sequentially because they are small, some filetypes have internal structure that can be used to infer related blocks and their order, and even finding a single block of a file can be useful if it happens to contain a spreadsheet row starting "facilitating payments to foreign officials" or, I don't know, "Fiat@".

You end up doing this kind of thing a lot because of a detail of file systems that all of my readers probably know. It's often articulated as something like "when you delete a file, it's not deleted, just marked as having been deleted." That's not exactly wrong but it's also an oversimplification in a way that makes it more difficult to understand why that is the case. There's a whole level of indirection due to block allocation, updating the bitmap on every file delete is a relatively time-consuming process that offers little value, actually overwriting blocks would be even more time consuming with even less value, etc. Read Brian Carrier for the whole story.

Actually, screw Brian Carrier, I've written before about the adjacent topic of secure erasure of computer media.

My point is this: these forensic methods are performed on a fully functional storage device (or more likely an image of one), where "recovery" is necessary and possible because of the design of the file system. The storage device, as hardware, is not all that involved. Well, that's really an oversimplification, and points to an important consideration in modern data recovery: storage devices have gotten tremendously more complex, and that's especially true of SSDs.

Even HDDs tend to have their own thoughts and feelings. They can have a great deal of internal logic dedicated to maintaining the disk surface, optimizing performance, working around physical defects on the surface, caching, encryption, etc. Pretty much all of this is proprietary to the manufacturer, undocumented, and largely a mystery to the person performing recovery. Thinking of the device as a "sequence of bytes" throws out a lot of what's really going on, but it's a necessary compromise.

SSDs have gone even further. Flash storage is less durable than magnetic storage but also more flexible. It requires new optimizations to maximize life and facilitates optimizations for access time and speed. Some models of SSDs vary from each other only by their software configuration (this has long been suspected of some HDDs as well, but I have no particular insight into Western Digital color coding). Even worse for the forensic analyst, the TRIM command creates a whole new level of active management by the storage device: SSDs know which blocks are in use, allowing them to constantly remap blocks on the fly. It is impossible, without hardware reverse engineering techniques, to produce a true image of an SSD. You are always working with a "view" of the SSD mediated by its firmware.

So let's compare and contrast forensic analysis to consumer data recovery. The problem for most consumers is sort of the opposite: they didn't delete the file. If they could get the sequence of bytes off the storage device, they could just access the file through the file system. The problem is that the storage device is refusing to produce bytes at all, or it's producing the wrong ones.

Techniques like file carving are not entirely irrelevant to consumer data recovery because it's common for storage devices to fail only partially. There are different ways of referring to the physical geometry of HDDs, and besides, modern storage devices (HDDs and SSDs alike) abstract away their true geometry. Different file systems also use different terminology for their own internal system of mapping portions of the drive to logical objects. So while you'll find people say things like "bad cluster" and "bad sector," I'm just going to talk about blocks. The block is the smallest elementary unit by which your file system interacts with the device. The size of a block is typically 512B for smaller devices and 4k for larger devices.

A common failure mode for storage devices (although, it seems, not so much today) is the loss of a specific block: the platter is damaged, or some of the flash silicon fails, and a specific spot just won't read any more. The storage device can, and likely will, paper over this problem by moving the block to a different area in the storage medium. But, in the process, the contents of the block are probably lost. The new location just contains... whatever was there before [2]. Sometimes the bad block is in the middle of a file, and that sucks for that file. Sometimes the bad block is in the middle of a file system structure like the allocation table, and that sucks for all of the files.

More complicated file systems tend to incorporate precautionary measures against this kind of thing, so the blast radius is mostly limited to single files. For example, NTFS keeps a second copy of the allocation table as a backup. Journaling can also provide a second source of allocation data when the table is damaged.

Simpler file systems, like the venerable FAT, don't have any of these tricks. They are, after all, old and simple. But old age and simplicity gives FAT a "lowest common denominator" status that sees it widely used on removable devices. PhotoRec, while oriented towards the consumer data recovery application, is actually a file carving tool. It's no coincidence that it's called PhotoRec. Removable flash devices like SD cards have simple controllers and host simple file systems. They are, as a result, some of the most vulnerable devices to block failures that render intact files undiscoverable.

What about the cases where file isn't intact, though? Where the block that has become damaged is part of the file that we want? What about cases where a damaged head leaves an HDD unable to read an entire surface?

Well, the news isn't that great. Despite this being one of the most common types of consumer storage failure for a decade or more, and despite the enormous inventory of software that promises to help, your options are limited. A lot of the techniques that software packages used in these situations lack supporting research or are outright suspect. Let's start on solid ground, though, with the most obvious and probably safest option.

One of the problems you quickly encounter when working with a damaged storage device is the file system and operating system. File systems don't like damaged storage devices, and operating systems don't like file systems that refuse to give up a file they say exists. So you try to copy files off of the bad device and onto a good one using your daily-driver file browser, and it hits a block that won't read and gets stuck. Maybe it hangs almost indefinitely, maybe you get an obscure error and the copy operation stops. Your software is working against you.

One of the best options for data recovery from suspect devices is an open-source tool called ddrescue. ddrescue is very simple and substantially similar to dd. It has one critical trick up its sleeve: when reading a block fails, ddrescue retries a limited number of times and then moves on. With that little adaptation, you can recover all of the working blocks from a device and so likely recover all of the files but a few.

Besides, just retrying a few times has value. Especially on magnetic devices, the result of reading the surface can be influenced by small perturbances. An unreadable sector might be readable every once in a while. This doesn't seem to happen as much with SSDs due to the dynamics of flash storage and preemptive correction of weak or ambiguous values, but I'm sure it still happens every once in a while.

At the end of the day, though, this method still means accepting the loss of some data. Losing some data is better than losing all of it, but it might not be good enough. Isn't there anything we can do?

HDDs used to be different. For one, they used to be bigger. But there's more to it than that. Older hard drives used stepper motors to position the head stack, and so head positioning was absolute but subject to some mechanical error. Although this was rarely the case on the consumer market, early hard drives were sometimes sold entirely uninitialized, without the timing marks the controller used to determine sector positions. You had to use a special tool to get the drive to write them [3]. It was common for older drives to come with a report (often printed on the label) of known bad sectors to be kept in mind when formatting.

We now live in a different era. Head stacks are positioned by a magnetic coil based on servo feedback from the read head; mechanical error is virtually absent and positioning is no longer absolute but relative to the cylinder being read. Extensive low-level formatting is required but is handled completely internally by the controller. Controllers passively detect bad blocks and reallocate around them. Honestly, there's just not a lot you can do. There are too many levels of abstraction between even the ATA interface and the actual storage to do anything meaningful at the level of the magnetic surface. And all of this was pretty much true in the late '00s, even before SSDs took over.

So what about SpinRite?

SpinRite dates back to 1987 and is apparently still under development by its creator Steve Gibson. Gibson is an interesting figure, one of the "Tech Personalities" that contemporary media no longer creates (insert comment about decay in the interest of capitalism here). Think Robert Cringely or Leo Laporte, with whom Gibson happens to cohost a podcast. In my mind, Gibson is perhaps most notable for his work as an early security researcher, which had its misses but also had its hits. Through the whole thing he's run Gibson Research Corporation. GRC offers a variety of one-off web services, like a password generator (generated, erm, server-side) and something that displays the TLS fingerprint of a website you enter. There's a user-triggered port scanner called ShieldsUp, which might be interesting were it not for the fact that its port list seems limited to the Windows RPC mapper and some items of that type... things that were major concerns in the early '00s but rarely a practical problem today.

It's full of some gems. Consider the password generator...

> What makes these perfect and safe? Every one is completely random (maximum entropy) without any pattern, and the cryptographically-strong pseudo random number generator we use guarantees that no similar strings will ever be produced again. Also, because

> this page will only allow itself to be displayed over a snoop-proof and proxy-proof high-security SSL connection, and it is marked as having expired back in 1999, this page which was custom generated just now for you will not be cached or visible to anyone else. ... The "Techie Details" section at the end describes exactly how these super-strong maximum-entropy passwords are generated (to satisfy the uber-geek inside you).

You know I'm reading the Techie Details. They describe a straightforward approach using AES in CBC mode, fed by a counter and its own output. It's unremarkable except that just about any modern security professional would have paroxysms at the fact that he seems to have implemented it himself. Sure, there are better methods (like AES CTR), but this is the kind of thing where you shouldn't even really be using methods. "I read it from /dev/urandom" is a far more reassuring explanation than a block diagram of cryptographic primitives. /dev/urandom is a well-audited implementation, whatever is behind your block diagram is not. Besides, it's server side!

My point is not so much to criticize Gibson's technical expertise, although I certainly think you could, but to say that he doesn't seem to have updated his website in some time. A lot of little details like references to WEP and the fact that the PDFs are Corel Ventura output support this theory. By association, I suspect that GRC's flagship product, SpinRite, doesn't get a lot of active maintenance either.

Even back around 2007 when I first encountered SpinRite it was already a little questionable, and I remember a rough internet consensus of "it likely doesn't do anything but it probably doesn't hurt to try." A little research finds that "is SpinRite snake oil?" threads date back to the Usenet era. It doesn't help that Steve Gibson's writing is pervaded by a certain sort of... hucksterism. A sort of ceaseless self-promotion that internet users associate mostly with travel influencers selling courses about how to make money as a travel influencer.

But what does SpinRite even claim? After a charming disclaimer that GRC is opposed to software patents but nonetheless involved in "extensive ongoing patent acquisition" related to SpinRite, a document titled "SpinRite: What's Under the Hood" gives some details. It's undated but has metadata pointing at 1998. That's rather vintage I see several reasons to think that there have been few or no functional changes in SpinRite since that time.

SpinRite is a bootable tool based on FreeDOS. It originated as an interleaving tool, which I won't really explain because it's quite irrelevant to modern storage devices and really just a historic detail of SpinRite. It also "introduc[ed] the concept of non-destructive low-level reformatting," which I won't really explain because I don't know what it means, other than it seems to fall into the broad category of no one really knowing what "low level formatting" means. It's a particularly amusing example, because most modern software vendors use "low level formatting" to refer explicitly to a destructive process.

SpinRite "completely bypasses the system's motherboard BIOS software when used on any standard hard disk system." I assume this means that SpinRite directly issues ATA commands, which probably has some advantages, although the specific ones the document calls out seem specious.

In reference to SpinRite's data recovery features, we read that "The DynaStat system's statistical analysis capability frequently determines a sector's correct data even when the data could never be read correctly from the mass storage medium." This is what I remember as the key claim of SpinRite marketing over a decade ago: that SpinRite would attempt rereading a block a very large number of times and then determine on a bit-by-bit basis what the most likely value is. It seems reasonable on the surface, but it wouldn't make much sense with a drive with internal error correction. That's universal today but I'm not sure how long that's been true, presumably in the late '90s this was a better idea.

That's probably the high point of this document's credibility. Everything from there gets more suspect. It claims that SpinRite has a proprietary system that models the internal line coding used by "every existing proprietary" hard drive, an unlikely claim in 1998 and an impossible one today without a massive reverse engineering effort. Consider also "its second recovery strategy of deliberately wiggling the drive's heads." It seems to achieve this by issuing reads to cylinders on either side of the cylinder in question, but it's questionable if that would even work in principle on a modern drive. You must then consider the use of servo positioning on modern drives, which means that the head will likely oscillate around the target cylinder before settling on it anyway.

This gives the flavor of the central problem with SpinRite: it claims to perform sophisticated analysis at a very low level of the drive's operation, but it claims to do that with hard drives that intentionally abstract away all of their low level details.

A lot of the document reads, to modern eyes, like pure flimflam, written by someone who knew enough about HDDs to sound technical but not enough to really understand the implications of what they were saying. The thing is, though, this document is from '98 and the software was already a decade old at the time! The document does note that SpinRite 3.0 was a complete rewrite, but I suspect it was the last complete rewrite and probably carried a lot of its functionality over from the first two versions.

I think that SpinRite probably does implement the functionality that it claims and that those features might have been of some value in the late '80s and much of the '90s. Then technology moved on and SpinRite became irrelevant. Probably the only thing that SpinRite does of any value on a modern drive is just rewriting the entire addressable area, which gives the controller an opportunity to detect bad blocks and remap them. That should also happen in the course of normal operation, though, and even tools dedicated to that purpose (like the open-source badblocks) are becoming rather questionable in comparison to the preemptive capabilities of modern HDDs. This type of bad-block-detecting rewrite pass is probably only useful in pathological cases on older devices, but it's also the only real claim of the vast majority of modern "hard drive repair" software.

It seems a little mean-spirited to go after GRC for their old software, but they continue to promote it at a cost of $89. The FAQ tells us that "SpinRite is every bit as necessary today as it ever was maybe even more so since people store so much valuable personal 'media' data on today's massive drives." I resent the implication of the scare-quoted "media," Mr. Gibson, but what I do with my hard drives in my own home is none of your business.

The FAQ tells us "SpinRite is often credited with performing "true miracles" of data recovery," but is oddly silent on the topic of SSDs. Some dedicated Wikipedia editor rounded up a number of occasions on which Gibson said that SpinRite was of limited or no use with SSDs, and yet the GRC website currently includes the heading "Amazingly effective for SSDs!" There is no technical explanation offered for how SpinRite's exceptionally platter-centric features affect an SSD, nor mention of any new functionality targeting flash storage. Instead, there's just anecdotal claims that SpinRite made SSDs faster and a suggestion that the reader google a well-known behavior of flash storage for which SSD controllers have considerable mitigations.

It is an odd detail of the GRC website that most of the new information about the product is provided in the form of video. Specifically, videos excerpted from recent episodes of Gibson and Laporte's podcast "Security Now." Security Now is weekly, so I don't think that SpinRite promotional material makes up a large portion of it, but it does seem conspicuous that Gibson uses the podcast as a platform for 15 minute stories about how SpinRite worked miracles. These segments, and their mentions of how SpinRite is a very powerful tool that one shouldn't run on SSDs too often, absolutely reek of the promotional techniques behind Orgone accumulators, Hulda Clark's "Zapper," and color therapy. It is, it seems, quack medicine for the hard drive.

I don't think SpinRite started as a scam, but I sure think it ended as one.

A lot of this was already apparent back in the late '00s, and I can't honestly say that bootleg copies of SpinRite every improved anything for me. So why did I love it so much? The animations!

SpinRite's TUI was truly a work of art. Just watch it go!.

[1] I recently bought the 20th anniversary vinyl box set of Meteora, which emphasizes that (1) 20 years have passed and (2) I am still a loser.

[2] This kind of visible failure seems uncommon with SSDs, likely because SSD controllers tend to read out the flash in a critical, suspicious way and take preemptive action when the physical state is less than perfectly clear cut. In a common type of engineering irony, the fact that flash storage is less reliable than magnetic media requires aggressive management of the problem that makes the overall system more reliable. Or at least that's what I tell myself when another SSD has gone completely unresponsive.

[3] Honestly this doesn't seem to have been typical with any hard drives by the microcomputer era, which makes perfect sense if you consider that these hard drives were sold with bad sector lists and therefore must have been factory tested. The whole "low level formatting" thing has been 70% a scam and 30% confusion with the very different technical tradition of magnetic diskettes, since probably 1990 at least.