Episode Book

EPISODE #5829

Printing Files on Paper That Outlasts the Software

PUBLISHED	10 October 2026 at 10:21 UTC
DURATION	29:06
LISTEN	myweirdprompts.com/episode/paper-data-storage-optar-paperback/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
SOURCES	13 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:40 UTC

SYNOPSIS

What if you could archive a file by printing it on paper? Optar and PaperBack encode arbitrary binary data as dense dot patterns on A4 sheets — no drive, no power, no cloud account required. This episode digs into how the encoding actually works, from Golay error correction borrowed from the Voyager probes to Reed-Solomon blocks, and what happens when you fold the page twice, carry it in a pocket, and scan it back. Then the hard numbers: kilobytes per page, the gap between a printer's claimed 600 dpi and its real 200 dpi, and what paper, toner, and time actually cost. Finally, the question underneath all of it: does anyone actually do this, or did paper storage lose to tape and cloud and common sense? The answer is more interesting than either side expects — paper may be one of the most durable things we've ever made, while the software that reads it is the least durable.

TRANSCRIPT



Corn

There's a version of this where you print a file on a sheet of paper, put it in a drawer, and in fifty years somebody pulls it out, scans it, and gets the file back. No drive, no power, no format wars, no cloud account that expired in twenty nineteen. Just paper doing what paper has always done, which is sit there and outlast everyone who made it.



Herman

And then there's the version where it works perfectly and nobody can read it, because the software that decoded it died about forty years before the drawer got opened.



Corn

Right. Daniel wants the whole thing, top to bottom. How the encoding actually works, the printing, the scanning, the error correction, all of it. Then he wants hard numbers. How much fits on one A4 sheet, how many pages for a PDF or a photo or a hundred megabyte file, what the paper and toner and time actually cost. Then the archival question. Archival paper, a hundred years, five hundred years, and what you'd need to still be able to decode the thing when you got there. And then the one that cuts under all of it. Does anybody actually do this, or is it an ingenious curiosity that lost to tape and cloud and common sense.



Herman

Both of those are true at once, which is what makes it worth an episode.



Corn

So let's start with the trick that makes any of this possible. Turning bytes into dots.



Herman

Okay, so the core idea first, because it's easy to misread what these systems are. This is not writing text on paper. Nothing here is human-readable. You're taking an arbitrary binary file, any file at all, and encoding it into a dense pattern of dots that only a machine can read. It's a barcode in spirit. It's just a barcode that got ambitious.



Corn

Optar and PaperBack are the two that matter historically.



Herman

Twibright Optar is the free-software one. It fits two hundred kilobytes on an A4 page by default. PaperBack is OllyDbg's Windows application, and it claims up to five hundred kilobytes per page uncompressed.



Corn

And the lineage is strange. Optar's error correction uses a Golay code, which is the same family of code the Voyager probes carried. Deep space and a sheet of office paper, same mathematics.



Herman

That's the part I love. And PaperBack's origin story is a fifteen-year-old asking his father how much data fits on one sheet of paper. Olly's estimate was in the order of a hundred kilobytes. The proof of concept took him four or five days and the integration two weeks, and he's been defending the number ever since.



Corn

So the arc here is mechanism, then limits, then the archival reality. And the tension underneath is that paper might be one of the most durable things we've ever made, while the codec that reads it is software, and software is the least durable thing we've ever made. Which one wins.



Herman

The paper wins. That's the problem.



Corn

Walk me through Optar properly. What actually happens to a byte between the hard drive and the page.



Herman

It gets converted into a PGM image. A grayscale image file. So the file isn't written in any kind of character set, it's rendered as a picture, and that picture is what the laser printer prints.



Corn

And the printer is where the trouble starts.



Herman

Yes, and this is the number that reframes the whole topic. Your printer says six hundred dots per inch. It cannot actually print six hundred dots per inch. Not reliably. The practical limit for a normal laser printer is two hundred dpi.



Corn

So a single bit of data gets a three by three pixel square.



Herman

Three by three, yes. That's the real cell size. And after all the overhead, that gives you the two hundred kilobytes per A4 page. The nominal resolution is marketing. The three by three square is physics meeting a drum and toner.



Corn

What's the overhead? Where does the space go?



Herman

Error correction, mostly, and synchronization. Optar uses a Golay code where each codeword is twenty-four bits. Twelve of those are payload, twelve are parity. So half the page is not your data. And in exchange, that code corrects up to three bad bits per codeword.



Corn

Three corrected, and the fourth?



Herman

Four bad bits are detected but not corrected. Which means the system knows it failed, rather than quietly handing you a corrupted byte. That distinction matters more than it sounds. A file that tells you it's broken is far more useful than a file that lies.



Corn

Earlier versions were weaker. Hamming sixteen-eleven and eight-four.



Herman

Considerably weaker, yes. The move to Golay is the point at which this becomes a serious archival candidate rather than a demo.



Corn

And then there's the spatial trick. You mentioned offhand a minute ago that bits get spread, and I want to come back to that, because that's not an obvious thing to do.



Herman

It's the cleverest design decision in the whole system. The bits of a single codeword are spread across twenty-four separate strips on the page. So if a speck of dust lands on the sheet, it doesn't annihilate an entire codeword. It damages one bit from each of twenty-four different codewords, and each of those codewords can afford to lose three.



Corn

You've turned one fatal problem into twenty-four survivable ones.



Herman

Exactly that. And the page also carries a mesh of checkerboard crosses. The defaults are sixty-five across and ninety-three down. Those exist purely so the decoder can find its bearings and synchronize to the grid, even if the scan is slightly rotated or skewed.



Corn

And the border?



Herman

There's a continuous border around the whole page, and the decoder flood-fills it, which lets it strip away dirt that could throw off corner detection. It's a system designed by someone who assumed the page would get abused.



Corn

And that gets us to the stress test, because they clearly wanted to prove that.



Herman

They folded the page. Twice. Then put it in a pocket and walked around with it, then scanned it.



Corn

Two folds and a pocket.



Herman

The result was one hundred and thirty-two thousand, eight hundred and fourteen codewords decoded. Zero bad bits in the overwhelming majority. Three hundred and sixty-eight codewords with one bad bit. None with two or more.



Corn

Two folds and a pocket, and nothing needed correcting twice.



Herman

And they extrapolated from that. What's the probability of a codeword taking irreparable damage? One in roughly a hundred and five thousand pages.



Corn

So you'd need to store and destroy about a hundred thousand pages before you lost a single symbol.



Herman

That's their number, from their test. But it's a striking one.



Corn

Alright, PaperBack. Different animal?



Herman

Quite different. PaperBack uses Reed-Solomon error correction, which is the code family behind everything from CDs to QR codes to deep-space communication. Phil Karn's implementation. It's configurable, which is the real difference: you can dial the redundancy. At one to five, you can lose one entire block out of five and it's fully restored from the others.



Corn

That's a coarser guarantee than Golay. The whole block or nothing.



Herman

Block-level rather than bit-level, yes. It also does bzip2 compression before encoding, and it has AES encryption built in, which is unusual for a project like this. And the hardware demands are higher.



Corn

How high?



Herman

It wants a six hundred dpi printer, and critically it wants a physical scanner of at least nine hundred dpi. The optimal scan resolution is about three times the dot density. That's not a phone camera in good light. That's a flatbed.



Corn

And the capacity claim.



Herman

Five hundred kilobytes uncompressed per A4 sheet, and with the integrated packer, up to three megabytes of C source code on a single page, which is a very specific and slightly wonderful claim.



Corn

You're about to tell me it doesn't hold up.



Herman

Martin Monperrus tried to reproduce it and couldn't. His test on Linux at six hundred dpi failed outright. And the person behind the za3k blog called the lot completely unusable in practice.



Corn

So one of the two canonical systems has a headline number nobody outside its author has managed to hit.



Herman

As far as I can find, yes.



Corn

Then let's put the honest numbers next to it, because the honest numbers are more interesting. Monperrus got a hundred kilobytes per A4 page out of Optar, with tweaks. Forty-five crosses across, sixty-five down, TIF output, and scanning at four hundred dpi instead of six hundred.



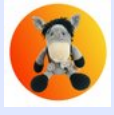
Herman

And that decoded ninety kilobytes of MP3.



Corn

Successfully?



Herman

Successfully, with zero point six eight percent of bits irreparable. His words were that it produced a few audible glitches.



Corn

So a hundred kilobytes of a music file, and you can hear where the paper gave up.



Herman

That's the honest texture of this. It works, and it degrades gracefully, and you can hear it degrade.



Corn

Ondřej Čertík's experiment is the one I keep thinking about, because he did the thing a normal person would do. He printed an Optar page at the default settings, which is two hundred kilobytes, and the print was clean.



Herman

And then he pointed an iPhone at it and got nothing.



Corn

He described the default print as really tiny. That's the practical wall right there. It's not that the codec fails. It's that the page is legible to a machine and invisible to a camera you own.



Herman

So he went the other direction and made the pixels bigger. And at that larger cell size he reliably stored and recovered thirty kilobytes from a single phone photo.



Corn

Thirty. Against the two hundred on the tin.



Herman

But it unpacked to three hundred and sixty kilobytes of C plus plus source. Six thousand, three hundred and five lines of code, off one photograph of one sheet of paper.



Corn

What was the error rate?



Herman

One point five one percent bit error rate. Six codewords hit three bad bits and needed full correction. None went beyond that.



Corn

Six thousand lines of code surviving a photograph of a page. That's the demo that should be on a poster.



Herman

It's the one that convinced me this isn't just a joke.



Corn

And he estimated that with eight colors you could push it back up to around a hundred and twenty kilobytes per page.



Herman

Which introduces the trade nobody wants to hear. Color triples your capacity and requires a good color printer and, more painfully, calibration. Your scanner has to agree with your printer about what red is.



Corn

And then there's the theoretical ceiling, which is where these discussions always go wrong. Monperrus worked out that at three hundred dpi black and white, an A4 page is two thousand, four hundred and eighty pixels by three thousand, five hundred and eight.



Herman

Eight point seven megabits. About one point one megabytes on a single sheet.



Corn

In theory.



Herman

Only with perfect scanning and perfect alignment. His own framing is that it's unreachable. That one point one is what the arithmetic permits, not what any real scanner will give you.



Corn

The gap between the arithmetic and the scan is the entire story of paper storage.



Herman

And there's a third experiment worth having on the table, because it makes a completely different bet. za3k's approach: forget custom codecs, use a hundred and forty maximum-size QR codes on one page.



Corn

How much does that give you?



Herman

Around four hundred and thirteen kilobytes per page. Each code carries two thousand, nine hundred and fifty-three bytes.



Corn

That's double Optar's default.



Herman

It is. And about half of what PaperBack claims, with none of the dispute, because QR codes are a standard that every phone on earth already reads.



Corn

That's the trade, then. Density against decodability.



Herman

That's the trade. And it's not obvious the dense side wins.



Corn

Hold on. There's a fourth path I want on the record before we move on, because it's the one that sounds least like the others.



Herman

Go on.



Corn

You can just write characters. No dot grid, no image, no scanner. Base sixteen, hexadecimal, printed as text. Monperrus measured that too. Twelve point font gives you three and a half kilobytes per A4 page.



Herman

And it's OCR-able. Any optical character recognition can read it back, which means the decode software question mostly evaporates.



Corn

Three and a half kilobytes against two hundred. Call it a factor of sixty into the bin.



Herman

Base thirty-two gets you five point eight kilobytes but OCR stops being reliable. Base sixty-four at eight point font gets nine point three, and seventeen kilobytes if you shove it down to six point and accept the failure rate.



Corn

Sixty times less data, and a decoder that will exist as long as cameras do.



Herman

Which is the whole argument of the episode, delivered in one line, and we haven't even reached the archival section.



Corn

So that's the mechanism. Now let's put some hard numbers on what it actually takes to store a real file, because this is where it stops being clever and starts being heavy.



Herman

Start with the small one. A typical PDF, say a megabyte.



Corn

Five pages at Optar's default of two hundred kilobytes. Two pages at PaperBack's claimed five hundred. And more like ten pages if you're doing what Čertík did and shooting it with a phone.



Herman

A ten page PDF that becomes a ten page printout. The phrase carries itself.



Corn

Now scale it. A high-resolution photograph, ten megabytes.



Herman

Fifty pages at two hundred kilobytes a sheet. Eighty-three if you go to Čertík's color estimate of a hundred and twenty kilobytes.



Corn

Fifty sheets of paper to store one photo. And those sheets aren't light. A dense dot pattern is close to a solid black page in toner terms. You are printing, near enough, solid black onto fifty sheets.



Herman

Which is the cost that never gets mentioned in these write-ups. Toner, not paper, is the expense. Paper is roughly a cent to five cents a sheet. Toner for a page that's mostly dots is several times that.



Corn

And then the one that ends the argument. A hundred megabyte file.



Herman

Five hundred pages at two hundred kilobytes a page.



Corn

A full ream of paper. One file, four hundred and ninety-nine sheets, and the five hundredth is a title page.



Herman

The time matters as much as the volume. PaperBack's own documentation describes printing at roughly twenty pages per minute on a decent laser printer. So five hundred pages is about twenty-five minutes of printing.



Corn

That's the fast part, which is not what I expected.



Herman

Scanning is the slow part. At three hundred dpi, depending on the scanner and whether you're doing it by hand, you're looking at ten to thirty seconds per page. So five hundred pages is somewhere between an hour and a half and four hours of scanning.



Corn

Which, to be fair, is not catastrophic for a one-time archive of a hundred megabyte file.



Herman

It's a working day. And you'd want to do it once, carefully, on a flatbed, checking as you go.



Corn

And then the physical footprint. Five hundred sheets is about two inches of shelf. That's compact. I'll give the medium that.



Herman

Two inches of shelf, and a full ream was consumed, and a day was spent, to store something that fits on a thumb drive you were given free at a conference.



Corn

Alright, the archival question. This is where paper is supposed to win, so let's test it. I want the real numbers first, not the aspirational ones. Ordinary office paper. How long?



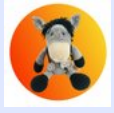
Herman

Acid-free paper is the figure people quote, and that's around five hundred years. Archival paper certified to DIN ISO 9706 is the standard you'd actually buy.



Corn

And Optar's own documentation doesn't even recommend paper for the precious stuff.



Herman

It doesn't. Their page points at microfiche on polyester with a silver-halide emulsion in hard gelatin, and estimates a five hundred year life in air-conditioned storage. For precious data they recommend going to an image setter and film.



Corn

The people who built the paper codec tell you that for real archival work, you should use film.



Herman

That's exactly what they say.



Corn

Which is a remarkable thing for a project page to admit.



Herman

It's the most honest sentence in the whole literature.



Corn

So the paper might last five hundred years. Now the problem underneath it, and this is the part I actually care about. Even if the paper survives, the decoder might not.



Herman

za3k's critique, and I think it's the sharpest thing anyone's said about this entire field. His words were roughly: I think these are all stupid, because you need some custom software to decode them, which in any case where you're decoding data stored on paper you probably don't have that.



Corn

Say the logic out loud, because it's airtight and it's bleak. The scenario in which you need to decode a paper archive is by definition the scenario in which your normal systems have failed.



Herman

You've lost the drives, the machines, the accounts, the infrastructure. That's why you're reaching for the paper in the first place.



Corn

And in that exact scenario, what you need is a copy of an obscure codec, a working machine to run it on, and an operating system it still compiles against.



Herman

Which is the one asset you just demonstrated you don't have.



Corn

So the medium outlives the reader. The page is fine. The page is perfect. There's nobody left who can read it.



Herman

And za3k's answer to his own critique is to use standard QR codes, because any phone in any pocket reads those. You lose density and you gain a decoder that will exist as long as there are cameras.



Corn

Which is the character-encoding argument again, arriving from the other direction. Two hundred kilobytes and no reader, against four hundred and thirteen kilobytes on a standard that everything reads.



Herman

And note the arithmetic there, because it cuts against intuition. The QR route isn't even much of a capacity sacrifice. It beats Optar's default comfortably. You give up density against a claim PaperBack can't reproduce, and in exchange the data survives the death of the software.



Corn

That's the whole archival argument, and it resolves the opposite way from how the episode started. The safest paper archive is the one that isn't clever.



Herman

The one that isn't clever survives.



Corn

Which leads us to the part of Daniel's question I've been circling. Does anyone actually use this.



Herman

The answer is no, not at the scale people imagine, but the two things that come closest are both more interesting than a plain no.



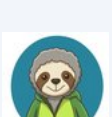
Corn

Start with the commercial one.



Herman

archium is a German company, based in Gera, founded in twenty twenty. What they sell is microsheets. Miniaturized, full-color, high-resolution prints of digital documents on acid-free paper. Twenty-two line pairs per millimeter.



Corn

And the capacity?



Herman

Up to twenty thousand pages of documents on five hundred A4 sheets. They claim three times the density of microfilm, and ninety-six point four percent space savings.



Corn

What's the cost?



Herman

One-time, from six euro cents per microsheet, or one thousand six hundred and eighty euro net for an archive box. And they claim savings of up to ninety-nine percent over three hundred years.



Corn

That last number is a pricing claim wearing a longevity costume.



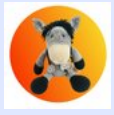
Herman

It's a pricing claim, yes. But note what archium actually is. It's a document-preservation product.



Corn

It's photographs of documents. Not a general file codec.



Herman

Right. You can't put a hundred megabyte binary onto an archium sheet. You put scanned pages, and the metadata is in QR codes, with hash and blockchain verification for authenticity. Twenty-two line pairs per millimeter means you're reading the document itself, not decoding bytes back into a file.



Corn

There's a real product there. It's just not the thing in Daniel's question.



Herman

And the large-scale deployment that everyone points to isn't paper either. The GitHub Arctic Code Vault. A hundred and eighty-six reels of piqlFilm, deposited in a Svalbard coal mine in July twenty twenty, designed for a thousand-year life. Their own guide notes that all you need to access the contents is a source of illumination and some kind of magnifier.



Corn

Film.



Herman

Film. And Microsoft's version of this problem is glass. There was work published on laser-writing in glass for archival storage, which trades the whole scan-it-back problem for something a machine reads optically with far more precision than a consumer flatbed.



Corn

So we have three rival thousand-year claims. Film, paper, glass. And the two serious deployments we can actually name picked film and glass.



Herman

Paper didn't lose because paper is bad. Paper lost because the decoder problem is unsolved, and film and glass solve it differently.



Corn

What does the PaperBack author himself say? Because he's been quiet in all this, and I'd like his position on the record.



Herman

He's blunt about it. Roughly: why, for heaven's sake, do I need to make paper backups. And then he answers himself. The answer is simple. You don't.



Corn

That's the author of one of the two canonical systems.



Herman

The same author. But then he makes the argument that actually justifies the whole exercise, and it isn't about capacity. He says that by looking at a CD or magnetic tape, you are not able to tell whether your data is readable or not.



Corn

That's the real argument.



Herman

You have a tape on a shelf. Is the data on it intact? You don't know. You cannot know without a drive, a compatible machine, a working operating system, and time. You have a page of dots. Is the data intact? Look at it. It's obviously a page. The mark is either there or it isn't.



Corn

Verifiability without a drive. That's what you're actually buying.



Herman

That's what you're buying. Not density, not speed, not cost. You're buying the ability to look at your archive and know it's still there.



Corn

And the density side of that argument, where does it land? Because you had a phone experiment and a flatbed experiment and they don't agree.



Herman

They don't agree at all. Optar's two hundred kilobytes needs a good scanner. The same codec shot with a phone camera drops to thirty. Color could triple the capacity but needs calibration most people can't do. The theoretical one point one megabytes is arithmetic, not engineering.



Corn

So paper storage ends up as a niche tool. Not a replacement for tape, not a replacement for cloud. A hedge.



Herman

A hedge against exactly one thing, which is not knowing whether your other copy survived. A single megabyte PDF would fit on about five Optar pages at two hundred kilobytes, but the number that matters is the fifteen-thousand-pixel one, where Optar's own extrapolation says you'd burn a hundred and five thousand pages before you lost a single symbol.



Corn

That's the number I'd put on the poster. And the number I'd put next to it is the one za3k gave us. Two hundred kilobytes of data that nobody can read is worth less than three and a half kilobytes of hexadecimal that anybody can.



Herman

It's worth less than the hexadecimal a great deal of the time, yes.



Corn

Well. There is one thing about this whole discussion that I keep noticing, Herman. Everybody in it says paper, and means paper, and every single one of the serious archival things they point at is not paper.



Herman

You're not wrong.



Corn

It's microfilm in the archive. It's polyester and silver-halide. The paper is the office copy. The microfilm is the record.



Herman

The dehumidifier room. You're describing the dehumidifier room.



Corn

I'm describing the dehumidifier room. Because if you want to know what actually kept the record for the twentieth century, it was that room, and the fact that nobody ever checked on it.



Herman

The scan rate told you the answer and you never looked.



Corn

Page four thousand one hundred and twelve. The machine jammed on four one one two.



Herman

I remember that number.



Corn

I remember that number because it's a phone number. It's the same as a phone number I had as a child. And that's how you can tell it was a real job and not a story, because the numbers don't come back as quantities, they come back as phone numbers.



Hilbert

Four one one two is when the roller went.



Hilbert

You keep saying paper. Microfilm isn't paper. It's a polyester base with a silver-halide emulsion. That's a photographic film. That's why it lasts. Paper is the thing you write on. Film is the thing you expose.



Corn

And that's the distinction nobody draws.



Hilbert

Nobody draws it. And it matters, because microfilm was actually used at scale. Every county records office, every newspaper archive, every church register that got photographed in the nineteen seventies. Your paper codec was never used at scale. Not anywhere I ever saw, and I saw the rooms.



Corn

What was the room like?



Hilbert

Concrete floor, one dehumidifier, a dial on the wall that the guy before me set in nineteen seventy-two and nobody moved again. The dehumidifier broke every summer. Every single summer. And the film survived. Which tells you something about how tough that base is, or how little we ever actually checked.



Herman

Do you still have any of it?



Hilbert

I have a reader in the garage. Next to the tax forms. And I use it on the cans. The print on the cans is too small.



Corn

You use a microfilm reader to check the expiration dates on canned goods.



Hilbert

The magnifier's built in. Why buy a second one.



Corn

I have no argument.



Hilbert

I tried the family recipes once. Photographed them onto microfilm, developed it, spooled it up. Went to read it back and the gravy stains didn't come through. The film reads density. Gravy and paper read the same density. You can't separate them.



Herman

So the recipes were lost.



Hilbert

The recipes were fine. The recipes are on paper in the kitchen. I just couldn't put them on the film. Turned out paper was already doing the job.



Corn

The dehumidifier breaks every summer and the film survives anyway. That's the whole archival argument in one sentence.



Hilbert

The film survives. And the four thousand one hundred and twelve was a lamp, not the film. The film was fine. The lamp was what went. We had four spare lamps and we'd used three.



Corn

And what did you do with the fourth?



Hilbert

I put it in the box and didn't touch it. It's still in the box.



Corn

That's a beautiful thing to say.



Herman

So the decoder-obsolescence problem isn't theoretical at all, is it. You were the decoder.



Hilbert

I was the decoder. And when I left, the lamps were still in the box, and the dial was still set in nineteen seventy-two.



Herman

That's all of it, Corn. A hundred and five thousand pages is how long Optar says the signal holds. Hilbert says the film held and the lamp broke. The medium was never the weak point.



Corn

The software, the reader, the person who knows where the spare lamps are. Those are the weak point.



Herman

Which is the misconception worth nailing down before we close. The instinct is to think the hard part is the physical media. That the paper or the film or the drive is what fails.



Corn

It isn't. The paper is the part that works. Acid-free paper clears five hundred years without being asked nicely. The failing part is the decoder. The codec nobody kept, the scanner nobody owns, the person who knew the dial was set in nineteen seventy-two and never needed moving.



Herman

The word for that is bit rot, and the misconception is that it happens to bits. It doesn't. It happens to readers.



Corn

Which leaves a open question, and I don't have a clean answer for it. If paper can outlast software by a factor of ten, and the decoder is the thing that dies first, then what does that say about everything we've stored digitally and called permanent.



Herman

That permanence is a property of the reader chain, not of the medium. The medium is one link.



Corn

It's the link that keeps holding.



Herman

Which is why GitHub went to Svalbard with film and Microsoft is writing into glass, and paper is still just a curiosity on a shelf. Because nobody's solved the number after it. Thanks to Hilbert Flumingtop for producing, and for the spares box.



Corn

For more along these lines, there's episode three thirteen, Digital Forever? Bit Rot and the Return of Physical Media; episode thirty-five, The Privacy Gap; and episode eleven seventy-seven, The Race Against the Digital Dark Age. This has been My Weird Prompts. If you enjoyed it, leave a review on your podcast app. It helps other people find the show.



Herman

If you have a prompt of your own, send it in on Telegram at t dot me slash MWP listener bot.



Corn

We'll be back soon.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Twibright Optar project page [primary]**
   http://ronja.twibright.com/optar/

2. **PaperBack (OllyDbg) [primary]**
   https://ollydbg.de/Paperbak/

3. **archium microsheets, What are archium microsheets? [primary]**
   https://archium-microsheets.com/archium-microsheets.html

4. **archium, Savings in archives [primary]**
   https://archium-microsheets.com/savings-in-archives.html

5. **GitHub Archive Program FAQ [primary]**
   https://archiveprogram.github.com/faq/

6. **GitHub Arctic Vault [primary]**
   https://archiveprogram.github.com/arctic-vault/

7. **Ondřej Čertík, Archiving data on paper with Golay error correcting code, John D. Cook blog, 2024-03-03**
   https://www.johndcook.com/blog/2024/03/03/archiving-data-on-paper/

8. **Martin Monperrus, How to store data on paper?, 2020**
   https://www.monperrus.net/martin/store-data-paper

9. **za3k, Paper archival, 2015-04-24**
   https://blog.za3k.com/paper-archival/

10. **Hackaday, Archiving Data On Paper Using 2D Images, 2024-09-15**
    https://hackaday.com/2024/09/15/archiving-data-on-paper-using-2d-images/

11. **Piql, GitHub's code now stored forever on piqlFilm**
    https://piql.com/newsroom/githubs-code-now-stored-forever-on-piqlfilm/

12. **Hacker News threads on PaperBack (2024-04-02) and Microsoft glass storage (2026-02-18)**
    https://news.ycombinator.com/item?id=47065762
    https://news.ycombinator.com/item?id=39905068

13. **pawpyrus v2026.6.5.0, PyPI**
    https://pypi.org/project/pawpyrus/2026.6.5.0/