Episode Book

EPISODE #5885

The World's Oldest Software Still Running

PUBLISHED	12 October 2026 at 09:04 UTC
DURATION	31:13
LISTEN	myweirdprompts.com/episode/oldest-software-still-running-26325/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	14 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:41 UTC

SYNOPSIS

Software is the only thing we build that we expect to throw away — yet some programs from the 1950s and 60s are still running the same jobs today. This episode digs into the strongest claims for the world's oldest production software: MOCAS, the 1958 Pentagon contract system; SABRE, the 1960 airline reservation system that cut booking time from 90 minutes to seconds; the IRS Individual Master File from 1962; and Voyager's 1977 flight computer, still taking commands from 22 light-hours away. Along the way, we untangle a distinction most coverage misses: a language still in use (COBOL) is not the same as software that has survived largely unchanged. Is this an engineering triumph or a warning sign? The answer depends entirely on which definition of "oldest" you're using — and whether the risk lives in the old code or in the rewrite.

TRANSCRIPT



Corn

Software is the only thing we build that we expect to throw away. You don't tear down a bridge because it's forty years old and the engineer who designed it has retired. The bridge is still holding traffic.



Herman

You repaint it. You inspect the cables. Nobody says, let's rebuild the Brooklyn Bridge in JavaScript.



Corn

Daniel's been thinking about exactly this. He wrote in asking about the world's oldest software still running in production. He suspects there are programs written in the fifties, sixties, seventies that are still doing the same jobs they were designed to do, and he wants to know where they are, how they keep running, who maintains them.



Herman

And what the oldest one actually is.



Corn

That's the part that gets slippery. Because he's also drawing a line I think most coverage misses. A language still in use isn't the same thing as software that has survived largely unchanged. COBOL is everywhere, but COBOL is a language.



Herman

Right, and there's a tension underneath all of it. Is this an engineering triumph or a warning sign? Because those are two very different stories about the same facts.



Corn

Let's start with the obvious place. The systems nobody got around to replacing because replacing them turned out to cost more than living with them.



Herman

So the framing that organizes everything here is that distinction Daniel drew. COBOL, 1959, is still enormously in production. Estimates range from two hundred and fifty billion lines to eight hundred billion lines, depending on who you ask and how they counted. That's a language, though. That's a language with a huge installed base.



Corn

Which is a category, not a time capsule.



Herman

A category, exactly. The interesting thing is the software. The actual surviving programs. And there the candidates sort themselves by which definition of oldest you're using.



Corn

MOCAS is the one that keeps coming up as the oldest claim. Pentagon system, 1958.



Herman

Nineteen fifty-eight. Department of Defense. Mechanization of Contract Administration Services. It tracked contracts and payments, and it was likely first written in FLOW-MATIC, which is Grace Hopper's language, before COBOL even existed. So MOCAS predates the language it eventually got associated with.



Corn

It's older than the standard.



Herman

Older than the standard. Then you have SABRE, which is the aviation entry. American Airlines with IBM. First experimental system online in 1960, fully operational by 1964 as the world's first fully operational computer reservations system.



Corn

And then the IRS, which is the government entry. Individual Master File, introduced 1962, 1963. And Voyager, which is 1977. Four candidates, and each one wins under a different definition.



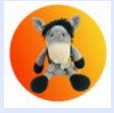
Herman

Guinness actually hedges on this. Their formal record says it's likely either SABRE or the IRS Individual Master File. They point out the IRS system is slightly newer but is also less likely to have been stripped of all its 1960s legacy code, which makes it maybe the more probable candidate.



Corn

So the record holder is a judgment call.



Herman

It's a judgment call, and the uncertainty isn't about whether these systems exist. It's about proving continuity from the earliest version to today. Because if you can't prove the code has been running the same job the whole time, you can't claim the record.



Corn

So let's take the strongest claim first. A Pentagon system from 1958 that's still cutting checks.



Herman

MOCAS is the one. MIT Technology Review reported on it in 2015, managing about one point three trillion dollars in obligations. Three hundred forty thousand contracts. Running on an IBM 2098 model E-10 mainframe with three hundred ninety-eight MIPS and eight gigabytes of RAM.



Corn

Three hundred ninety-eight MIPS.



Herman

Million instructions per second. That's the whole machine. And it's replacing efforts have repeatedly sputtered. Cost, complexity, transition planning. They've tried to move off it more than once.



Corn

What does eight gigabytes do in a Pentagon contract system? That's less than a phone.



Herman

Less than a phone, and it's been moving a trillion dollars in obligations. That's the thing that gets me. The scale of the consequence is enormous and the machine underneath is tiny by any modern measure.



Corn

So MOCAS is the oldest claim. SABRE is the cleanest story.



Herman

SABRE is the one with the best documentation of what it actually did. Developed by American Airlines with IBM. Ran on two IBM 7090s. Handled seventy-five hundred reservations per hour. And this one's my favorite number in the whole episode, it cut booking time from ninety minutes to seconds.



Corn

Ninety minutes per booking.



Herman

Per booking. You'd sit with an agent and a phone and a card index, and it took an hour and a half to confirm a seat. SABRE made it instantaneous. That's not an efficiency gain, that's a different business.



Corn

That's the moment the airline industry stopped being a paperwork operation.



Herman

And it didn't stay on those 7090s. It migrated to PARS and System 360 in the early seventies, then its operating system evolved into ACP, and then into TPF in 1979. Transaction Processing Facility. Which is the bit that matters, because TPF didn't stay a Sabre thing. It became the operating environment for the whole category. IBM says almost all the biggest banks, insurers, retailers and airlines still use it.



Corn

So SABRE isn't just a survivor. It's the ancestor of an entire industry's plumbing.



Herman

The ancestor of the plumbing. Amadeus was founded in 1987 and built on TPF. Galileo, which became Travelport, same thing. And between Sabre, Amadeus and Travelport, they handle essentially every airline booking on Earth.



Corn

Every seat you've ever booked went through something descended from a 1960 reservation system.



Herman

In some sense, yes. The transaction semantics, the way a seat gets locked and sold, that lineage runs straight back.



Corn

Now the IRS one is the one that feels the most alive to me. Because 1962 is not ancient history in the way 1958 feels ancient. Nineteen sixty-two is recent enough that people you know were alive.



Herman

The Individual Master File. Introduced 1962, 1963. Written in a mix of COBOL and IBM Assembler. And as much as twenty million lines of it are reportedly Assembler. That's the key detail, because Assembler tied to System 360 architecture can't run on anything but an IBM mainframe.



Corn

It's welded to the architecture.



Herman

Welded to the architecture. You can't lift it and drop it somewhere else. The IRS has been trying to modernize since at least 2009. The watchdog calls it one of the most complex modernization programs in the federal government. And Tax Notes reported in March of 2025 that the program is thirty-five years old and allegedly fifteen billion dollars over budget.



Corn

Thirty-five years of trying to replace a thing.



Herman

Thirty-five years. And Commissioner Danny Werfel said in May of 2024 that the agency is testing a modernized Individual Master File, targeting fiscal 2028 retirement. He called it a proud moment.



Corn

A proud moment. In 2028. For a system from 1962.



Herman

If it happens. Then Voyager, which is the edge case, because it's not a business system. It's a machine that left the solar system. Launched 1977, running 1970s assembly on about sixty-eight kilobytes of total memory.



Corn

Sixty-eight kilobytes.



Herman

Total. Not per processor. Total. Guinness records the Voyager Computer Command System as the longest continually operating computer, forty-three years and seventy days as of October of 2020. As of July of 2026, Voyager 1 is forty-eight years and ten months into its mission.



Corn

And it's still taking commands.



Herman

Still taking commands. NASA shut off its Low-Energy Charged Particles instrument in April of 2026, after almost forty-nine years, to save power. The radioisotope generators lose about four watts a year. So they're turning things off, one at a time, to keep the transmitter alive.



Corn

Four watts a year. That's the whole clock.



Herman

And the command round-trip is forty-five hours. Which means every change you send, you find out two days later whether you broke it.



Corn

Twenty-two and a half hours each way.



Herman

Each way. A Big Bang power-rebalancing change was canary-tested on Voyager 2 in May and June of 2026, with Voyager 1 scheduled for July. You test the risky change on the twin first, because that's the only way to find out.



Corn

There's no staging environment for a spacecraft.



Herman

There is no staging environment. There is one other spacecraft, and it's also forty-eight years old.



Corn

So that's the 1958, the 1960, the 1962, the 1977. What about the whole sector view? Banks aren't one system.



Herman

Banks are the most alarming picture in the research. Baringa surveyed two hundred UK banks, and the results were reported in September of 2025. Sixteen percent rely on software written in the 1960s. Almost forty percent on 1970s code. Half of them admitted relying on software that only one or two members of staff understand, and those staff are at or near retirement.



Corn

One or two people.



Herman

Thirty-eight banks still use code designed for punch cards. Fifteen percent run code written for room-sized mainframes. And the quote from Baringa's Paul Mihajlovic is the one that should be on a poster. He said, if you have a current account, your money could rely on code written when Harold Macmillan was prime minister.



Corn

Every direct debit in Britain is running on Macmillan-era logic.



Herman

The clearing system, the standing orders, the interest calculation, all of it. And then there are the anecdotal industrial examples. Christopher Roosen wrote a piece in July of 2025 about what he calls zombie software. The San Francisco Metro still runs its Automatic Train Control System off a DOS-loaded floppy disk.



Corn

A floppy disk moving trains.



Herman

A floppy disk. Deutsche Bahn posted a job ad seeking a sysadmin with Windows 3.11 and DOS experience. ATMs, sometimes, still run Windows 98. Now, I want to flag the evidentiary line here, because it matters. The mainframe stuff is well documented. The floppy disk and Windows 98 stuff is thinner. It's real, but it's harder to confirm at scale than something like the UK banking survey.



Corn

The floppy disk is the story people repeat.



Herman

It's the story people repeat, and it's probably true somewhere, but it's not the load-bearing evidence. The load-bearing evidence is a trillion dollars in Pentagon obligations and half of UK banking.



Corn

Here's the thing I want to be clear about, though. COBOL being everywhere is not the same as a 1968 program still running. Those are two different claims and the whole episode turns on which one you're making.



Herman

And New Scientist's investigation in March of 2025 drew that line sharply. What they found is that old code often doesn't survive as a whole system, it survives as snippets buried inside modern programs. Perl, which dates to 1987, still contains code from its first version. Linux's indent command uses code from 1976.



Corn

Nineteen seventy-six, inside a tool people run today without thinking.



Herman

Without thinking. Red Hat's Mike McGrath pointed that out. And then there's the mission software case, which is the one where the caution makes total sense. ESA's James Eggleston said it plainly. These things are billion-dollar satellites, so the people who fly them are really not keen on doing anything that's not completely guaranteed to work. If you mess it up, your spacecraft is fried. End of mission.



Corn

That's the whole argument in one sentence. Nobody rewrites the flight software because rewriting it is the risk.



Herman

The risk is the rewrite, not the old code. That reframes the whole question. We treat the old code as the problem, and from inside the operation, the old code is the thing that isn't broken.



Corn

What about telecom? Daniel specifically asked about telecommunications.



Herman

Honest answer, I don't have a named, dated telecom system for you. The sector gets cited generically, SS7, switch software, that whole world, but I couldn't find a specific system with a 1950s or 60s origin that's documented the way MOCAS or the IRS file is. So the telecom leg of Daniel's question is the one we can't fully deliver on.



Corn

That's worth saying out loud. The interesting question is why telecom is less documented than banking or aviation.



Herman

My guess, and it's only a guess, is that telecom operators are even less eager to talk about it than banks are. Which sets up the actual finding here, which isn't about any single system. It's about the silence.



Corn

The silence.



Herman

New Scientist went looking for comment. Sabre, Delta, American, United, Lufthansa, British Airways, Apple, Microsoft, HP, Texas Instruments, HSBC, Lloyds, NatWest, Barclays, Lloyd's, PayPal, Visa, Mastercard, NASA. All of them declined or failed to respond.



Corn

Every single one.



Herman

The trustee of the LEO Computers Society, Vincent Bodsworth, had the line that explains it. He said, they're probably nervous about it.



Corn

They're nervous because they don't know.



Herman

They're nervous because nobody fully knows what's running, and the answer would be embarrassing or alarming depending on who's asking.



Corn

So the strongest evidence for the scale of this is that the people who would know won't say.



Herman

And that's a strange thing to build an argument on, but I think it's the right read. If the systems were clean, a few of those companies would have sent a statement. The silence is data.



Corn

So we know what these systems are and where they live. The harder question is how they're still running. And who's keeping them alive.



Herman

The hardware question is where most people get it wrong. The mental image is a vacuum-tube machine in a basement, and that's not what's happening.



Corn

Almost never, right?



Herman

Almost never. MOCAS runs on an IBM 2098 model E-10. That's a modern mainframe. It's not a 1958 machine. The IRS file is tied to System 360 architecture but runs on current IBM mainframes. SABRE migrated from the 7090s to System 360 in the early seventies, then to TPF in 1979. The pattern is that the logic is ancient and the hardware is not.



Corn

So you're running 1958 logic on 2020s silicon.



Herman

Nineteen fifty-eight logic on modern silicon. And it's not just IBM's own boxes. Amadeus has retired its last mainframes to cloud. But the thing they preserved was the mainframe transaction semantics. They moved the hardware and kept the behavior.



Corn

That's the distinction. Nobody's running a 1958 computer. They're running 1958 decisions.



Herman

Nineteen fifty-eight decisions, on whatever the current machine is. Which is the real answer to Daniel's question about original hardware versus emulation versus modern mainframes. The honest answer is all three, in different proportions, and mostly the third one.



Corn

There's a pattern in how these get modernized that I want to get into. Because nobody replaces them. That's the thing that surprised me.



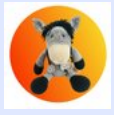
Herman

Nobody replaces them, and there's a name for the approach. It's called the strangler fig. You don't cut the old tree down. You grow around it. You wrap the old system in APIs and you extract the non-core functions, one at a time, until the old thing is doing less and less. But it stays. It stays until the new thing is proven.



Corn

And it's never proven.



Herman

It's never fully proven, because the new thing can't be proven until it's handled every edge case the old one accumulated over sixty years. And the edge cases are undocumented, because the documentation is in someone's head. So the old system runs in parallel with



Herman (continued)

the new one, potentially for years, because you can't turn off the old one until the new one is perfect, and the new one can't be perfect until it's seen everything the old one has seen.



Corn

That's why the modernization programs run for decades.



Herman

That's exactly why. The IRS is building the Individual Tax Processing Engine, and a next-gen version of the file called TAMI-I, converting COBOL and Assembly to Java. But the old system runs in parallel until the new one is proven. And this is a program that's been running for thirty-five years and is fifteen billion dollars over budget.



Corn

So the plan is to port sixty years of accumulated edge cases, and every edge case you miss is a tax return that gets processed wrong.



Herman

Every missed edge case is a real person's money. That's why it's slow. The slowness isn't incompetence, it's the price of being right.



Corn

Okay, but the thing that actually worries me is the people. Not the machines.



Herman

That's the real deadline, and it's the strongest finding in the research. Bill Hinshaw runs a firm called COBOL Cowboys. He's eighty-two. He's credited with writing the first bank ATM software in 1970. His youngest contractor is forty. Most of them are in their sixties and seventies. And he said, we don't see a problem with working on COBOL right now. Ten years from now, we may. That's the deadline. The hardware is fine. MOCAS is running on a modern mainframe. The IRS file runs on current IBM iron. The systems work. What doesn't



Herman (continued)

work is the supply of people who understand them.



Corn

And the Baringa number backs that up. Half of UK banks rely on software only one or two staff understand, and those staff are at or near retirement.



Herman

One or two people per bank. And when they go, the knowledge goes with them. This isn't theoretical. This is a survey of two hundred banks that produced those numbers.



Corn

So the systems are load-bearing, and the people holding them up are in their seventies.



Herman

And a lot of what they know isn't written down, because it was never worth writing down until the only person who knew it started thinking about retiring. That's the gap.



Corn

Which is where AI comes in.



Herman

Bloomberg reported in April of 2025 that the race to fix these aging systems is heating up with AI's help. The pitch is that a model can read COBOL and Assembler faster than any human, and translate it into something modern. And in December of 2025, eleven House Ways and Means Democrats demanded answers from Treasury about the IRS's use of AI on what they called antiquated COBOL code.



Corn

So both parties are watching the same tool.



Herman

Both are watching the same tool, for opposite reasons. And this is the part where let me be careful, because it cuts both ways. If the tool works, it can read and translate code that no living person fully understands. If it hallucinates, it's touching bank transactions and tax records.



Corn

It's the same model doing both.



Herman

It's the same model. The thing that might save these systems and the thing that might break them is the same capability. And the risk isn't the model being wrong in a vacuum. It's the model being wrong in a system where the humans checking it also don't fully understand the original.



Corn

If the person reviewing the AI's translation doesn't understand the COBOL, what are they reviewing?



Herman

They're reviewing the output against nothing. That's the danger. The whole value of AI here is that it knows something the humans don't. And the whole danger is that it knows something the humans can't verify.



Corn

Let me ask you something, because this is where I get stuck. Does the oldest question even have an answer? Or is it a definitional mess?



Herman

It's definitional, and I think that's the honest landing. MOCAS, 1958, is the oldest claimed. But Guinness's formal record points to SABRE, 1960, or the IRS file, 1962, 1963. And the reason isn't that MOCAS doesn't exist, it's that proving continuity is harder for MOCAS than for the other two.



Corn

And then there's the stuff that's older but doesn't count.



Herman

EDSAC code from the 1940s and 50s is among the oldest software ever written. But it won't run until the Bletchley Park replica is finished, so it's not in production. It's a museum piece. And then there's Sparkler Filters in Conroe, Texas.



Corn

The punchcard one.



Herman

Sparkler Filters still used a 1948 IBM 402 punchcard system for inventory and accounting as of 2015. But the IBM 402 has no memory. It ran programs wired onto plugboards. So it's arguably hardware, not software. The program was the wiring.



Corn

So the oldest thing might not even be software.



Herman

The oldest thing might not qualify as software at all, depending on how you define it. Which is the whole point. The definition determines the winner.



Corn

That's the answer to Daniel's question. There's no single answer, and that's not a cop-out, that's the finding.



Herman

The definition picks the champion. If you count language, COBOL wins by volume. If you count unchanged code, the IRS file. If you count continuous operation of a computer, Voyager. If you count oldest origin, MOCAS.



Corn

Four questions, four answers.



Herman

Four questions, four answers, and nobody is lying.



Corn

There's a line that I keep coming back to. Christopher Roosen, the Zombie Software piece. He wrote that legacy is not the code that time forgot. Legacy is the system that outlived its creators because it kept being worth saving.



Herman

Kept being worth saving. That's the reframe. These aren't fossils. They're doing a job that still needs doing, and they're doing it well enough that replacing them is the riskier move.



Corn

And Stuart Mackintosh from the Perl Foundation had the other half of it. On why Perl's old code is still there, he said, it just works.



Herman

And then he said something that sticks with me. We're in the very first flash of the earliest stage of the digital revolution, and we're going to build on this same infrastructure.



Corn

First flash.



Herman

When you look at it that way, a sixty-year-old system isn't old. It's the beginning. We're fifty, sixty years into a project that might run for centuries, and we're already calling things legacy that are, in the grand scheme, still warm.



Corn

So which is it? Triumph or dependency?



Herman

I land on both, and I don't think you can separate them. It's a triumph that these systems still work. It's a dependency that we can't replace them. The same fact produces both readings. That's why the question doesn't resolve.



Corn

We still haven't answered the question Daniel actually wanted answered, which is what the oldest one is. And the plain answer is we can't.



Herman

We can't, and the reason we can't is interesting. The question assumes we know what's running. And the silence from every major bank, airline and tech firm suggests we don't. You can't name the oldest system when the people running the systems won't tell you what they've got.



Corn

Which means the true oldest system might be something nobody's asked about yet.



Herman

Something nobody's asked about, running quietly, in a building with no windows, on a mainframe that just works.



Hilbert

Quick question, and I mean a real one. When you two say the systems work, what does that mean? Because I've been sitting here thinking about my cousin's husband.



Corn

Go on, Hilbert.



Hilbert

His name is Duane. He spent thirty-four years as a systems programmer for a regional insurance company out in Ohio. Claims processing. And his whole job, for three decades, was maintaining a system that was written in 1968.



Herman

Nineteen sixty-eight.



Hilbert

He retired in 2019. And the company called him back as a contractor four times in the first two years after he left.



Corn

Four times.



Hilbert

Because the system wasn't broken. That's my point. You don't replace your grandmother's cast-iron skillet because it's old. You replace it because it's broken. Duane's system wasn't broken.



Herman

So what was he doing all those years, if it wasn't broken?



Hilbert

He'd say the system had personality. It would occasionally process a claim in a slightly different order than you expected. Nothing wrong with the output. Just a different order. And the only way to know was to have worked with it long enough to feel it. He could hear when the system was about to have a bad day.



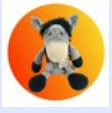
Corn

Hear it.



Hilbert

Not literally. But he'd walk past the operator console on a Tuesday morning and know. Some sound, some pattern in the batch log. After thirty years, you just know.



Herman

That's the undocumented knowledge problem in one man.



Hilbert

And here's the thing you'll like. He kept a notebook. A spiral-bound notebook. Handwritten.



Corn

What was in it?



Hilbert

Notes like, if the Tuesday batch runs long, check the tape drive before you call anyone. That kind of thing. Little rules. Two hundred pages of them, at the end. And the company never digitized it. Never asked for a copy.



Herman

They never asked.



Hilbert

Never asked. So when he came back as a contractor, he brought it with him. Because it was his. Nobody else on the floor had ever seen it.



Corn

What did they do when they saw it?



Hilbert

They asked him if he could make copies. He said no. He just said the notebook was his, and if they wanted it, they should have asked for it in 2005.



Corn

He held the institutional knowledge hostage with a spiral notebook.



Hilbert

He held it in his pocket for thirty years and they never asked. That's not hostage, that's just showing up.



Herman

Was there anything in the notebook that was more than operational notes?



Hilbert

There's a page in it. All caps. It says, DO NOT ASK ABOUT THE 1987 INCIDENT.



Corn

All caps.



Hilbert

Underlined twice. And when I asked him about it, he said, I can't talk about that. And changed the subject.



Herman

To what?



Hilbert

To the weather. He's not a weather guy. He changed the subject to the weather.



Corn

How long have you been trying to find out?



Hilbert

Seven years. I've asked his wife. I've asked two guys who used to work with him. I tracked down a man who worked in the mailroom in the eighties. Nobody will tell me. One of them said, you don't want to know. Which is exactly what someone says when you do want to know.



Herman

Is there anything else in the notebook about 1987?



Hilbert

There's a page before it that just says, if the batch has been running for more than eleven days, do not interrupt it. Do not shut it down. Do not call anyone. Just wait.



Corn

Eleven days.



Hilbert

And the page after the warning page has a name on it. Just a first name and a phone number. Area code 614.



Herman

Columbus, Ohio.



Hilbert

I called the number once. A woman answered. I said I was asking about a batch job. She hung up. I called back and it was disconnected. Same day. I checked the phone book, no listing. I checked the registry, nothing. Six weeks later a different number, same area code, listed to the same name.



Corn

So you did find something.



Hilbert

I found that somebody wants to not be found. Which I already suspected. And I have a theory about what happened, but it involves a hailstorm, and I'm not going to get into the hailstorm.



Herman

You're not going to say anything about the hailstorm.



Hilbert

Not on air. Not until I've confirmed the thing about the actuary.



Corn

There's an actuary.



Hilbert

There's always an actuary. That's how you know it's a real story. And I once spent six weeks trying to Google whether a regional Ohio insurance company had a batch job run for eleven days in 1987. Found nothing. Which, if you think about it, is proof.



Herman

Proof of what?



Hilbert

Proof that it happened. If it hadn't happened, there'd be a normal record. The absence is the evidence.



Corn

That's the same logic as the companies that won't talk.



Hilbert

Same logic. When nobody says anything, the nothing is the thing. That's how you know where the bodies are buried. Anyway, that's what I had. Herman, your levels are drifting on the left channel. Second half of the show. Come find me in the booth after.



Herman

Will do.



Corn

So here's what I'm stuck on. If the systems work, and the people who understand them are retiring, and AI is the proposed fix, what happens when the AI is trained on code that nobody fully understands?



Herman

Then you have a model that learned from a teacher who can't check its work. The model might get it exactly right. The model might reproduce a sixty-year-old bug that's been producing slightly wrong numbers that everybody adjusted for by hand and forgot to mention. Because the adjustment wasn't in the code, it was in the person.



Corn

Like Duane hearing the system have a bad day.



Herman

That adjustment doesn't survive the translation. It was never written down.



Corn

So the oldest-system question stays unresolved by design. MOCAS, SABRE, the IRS file, Voyager. Each one wins under a different definition, and the listener can decide which definition matters to them.



Herman

And the closing thought is that these systems aren't fossils. They're load-bearing. The question isn't whether we should replace them. It's whether we can, before the last person who understands them walks out the door with a notebook in his pocket.



Corn

If you enjoyed this one, leave us a review. It helps other people find the show.



Herman

Thanks to our producer, Hilbert Flumingtop, for the levels note and the story about the notebook, which is going to haunt me.



Corn

If you enjoyed this one, try episode four twenty-five, The Arc of Deprecation; episode four, If Your Voice Ages, Does Your Fine-Tune Become Useless; and episode one sixty-three, Big Iron. This has been My Weird Prompts. Send us your own prompt on Telegram at t dot me slash MWP listener bot. We'll be back soon.

SOURCES

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

1. **MIT Technology Review**
   What Is the Oldest Computer Program Still in Use? (2015-08-06)
   https://www.technologyreview.com/2015/08/06/166822/what-is-the-oldest-computer-program-still-in-use/

2. **Guinness World Records**
   Oldest software system in continuous use [primary record]
   https://www.guinnessworldrecords.com/world-records/636196-oldest-software-system-in-continuous-use

3. **Guinness World Records**
   Longest period of continual operation for a computer (Voyager CCS)
   https://www.guinnessworldrecords.com/world-records/635980-longest-period-of-continual-operation-for-a-computer

4. **New Scientist**
   The critical computer systems still relying on decades-old code (2025-03-05)
   https://www.newscientist.com/article/2470201-the-critical-computer-systems-still-relying-on-decades-old-code/

5. **Computer Weekly**
   UK banks still run software code written more than 60 years ago (2025-09-18)
   https://www.computerweekly.com/news/366631494/UK-banks-still-run-software-code-written-more-than-60-years-ago

6. **Nextgov/FCW**
   IRS is making headway on modernizing 1960s-era tax system (2024-05-20)
   https://www.nextgov.com/modernization/2024/05/irs-making-headway-modernizing-1960s-era-tax-system-commissioner-says/396695/

7. **Wikipedia**
   Sabre (travel reservation system), (travel_reservation_system)
   https://en.wikipedia.org/wiki/Sabre

8. **IN-COM Data Systems**
   Airline Reservation Systems: Why They're Still on Mainframes (2026-08-20)
   https://www.in-com.com/blog/airline-reservation-systems-why-theyre-still-on-mainframes/

9. **Christopher Roosen**
   Zombie Software Goes On and On and On (2025-07-30)
   https://www.christopherroosen.com/blog/2025/7/30/zombie-software

10. **Dominic D'Apice**
    The Oldest System in Production: 49 Years of Voyager 1 (2026-07-13)
    https://dominicdapice.com/blog/2026/07/the-oldest-system-in-production/

11. **Tax Notes**
    The Modernization Program That Would Not Die (2025-03-28)
    https://www.taxnotes.com/featured-analysis/modernization-program-would-not-die/2025/03/28/7rt5v

12. **FedScoop**
    House Democrats demand answers on IRS's AI use to manage taxpayer data (2025-12-18)
    https://fedscoop.com/irs-using-ai-cobol-doge-house-dems-letter/

13. **TechChannel**
    Special Report: COBOL Survey Results
    https://techchannel.com/cobol/special-report-cobol-survey-results-prove-pervasiveness-value-and-a-bright-future/

14. **Hacker News discussion**
    The oldest computer in use by the U.S. government
    https://news.ycombinator.com/item?id=12568829