Episode Book

EPISODE #5826

The Thread That Outlived Its Authors

PUBLISHED	10 October 2026 at 10:01 UTC
DURATION	24:36
LISTEN	myweirdprompts.com/episode/email-thread-outlives-authors/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:40 UTC

SYNOPSIS

In April 1997, Martin Schulze filed a wishlist bug against dpkg. In October 2021, Glenn Washburn quoted the entire original message and seconded it. That 24-year gap is the doorway into a bigger question: what happens when an email thread outlives the people who started it? This episode digs into the Debian bug tracker, the 39-year Traveller Mailing List archive, and the Cypherpunks archives, then pulls apart the machinery that holds threads together — Message-ID, In-Reply-To, and References — including the nineteen-year stretch when a reply header was legally allowed to read "35 ham and cheese sandwiches." Along the way: a threading algorithm designed to survive garbage input, a spec that deliberately throws away the middle of a conversation's ancestry, and the honest admission that nobody has written down a rule for replying to a dead man's message.

TRANSCRIPT



Corn

What if a conversation could outlive everyone who started it? That's the question Daniel left us with, and it's been rattling around my head all week.



Herman

It's a good one. It's a very good one.



Corn

So here's what he's asking. We've talked before about how stubborn mailing lists are, how they just keep going while everything around them burns down. And Daniel noticed something that follows from that. Some of these communities have been running so long that a single conversation inside them can plausibly outlast the people who were there when it started.



Herman

Which is a different thing from a long-lived list.



Corn

Right. And he's careful about that distinction. He doesn't want the oldest list. He wants the longest-running individual thread. One conversation, one chain of replies, persisting for an extraordinary length of time. He brings up a few examples from the Debian bug tracker. Bug #8639, filed in April of ninety ninety-seven, and then someone replies in October of twenty twenty-one to endorse the original request. Twenty-four years later. There's a second bug, #299307, that he says was closed with a reply pointing back at a message from two thousand five. And then the Cypherpunks archives, which he describes as having these incredibly intricate branching conversations going back to the nineties. He wants to know if we can find better examples, and whether those threads were continuously alive or just periodically resurrected.



Herman

And then the machinery.



Corn

How do Message-ID, In-Reply-To and References actually hold a thread together, and is there a ceiling on how deep or long one of these trees can get. And then the third part, which is the one that got me. What happens when a discussion survives its participants.



Herman

He called it slightly morbid.



Corn

He did. And then he called it fascinating, which I think is the more honest word. Conversations continuing after the people in them have died. Whether anyone has written down a rule for replying to a dead man's message.



Herman

That last part is where the research falls apart. I want to flag that early.



Corn

Then flag it.



Herman

Barely anything out there. We'll get to it, but I don't want anyone listening to think we're holding back a stack of documented cases. There isn't one.



Corn

Start with the flagship, then. The Debian bug.



Herman

Bug #8639. Filed the tenth of April, nineteen ninety-seven, by Martin Schulze. And it's a wishlist item, which matters. It's not a crash, it's a request. He wanted dpkg to be able to install, remove and purge a package even when the pre-inst and post-inst scripts fail. His stated reason is very human. He says he mistakenly broke some of his own installation scripts and ended up hacking around inside dpkg's database by hand, and he calls it ugly. That's the whole bug. A man annoyed at himself, filing a note so the tool would be better next time.



Corn

And then nothing for twenty-four years.



Herman

Not nothing. That's the thing. It was not asleep in a drawer. It was closed, reopened, retitled twice, in two thousand seven and again in twenty nineteen. In twenty fourteen it got merged with another bug, #752279. In twenty fifteen it received a spam message, a German job advertisement, which I find weirdly perfect. So it's this low-level hum of activity for a quarter of a century.



Corn

And then Glenn Washburn.



Herman

October fifteenth, twenty twenty-one. He quotes the entire original message, the whole thing from ninety-seven, and writes, twenty plus years later and I'd like to second this feature request. That's a twenty-four year, six month gap between the ask and the second.



Corn

There's something very funny about quoting the entire original message.



Herman

He's being polite. He's establishing the thread.



Corn

He's establishing the thread by reproducing twenty-four years of silence in one block quote.



Herman

It's how you do it. If you're replying into a thread that old, you re-anchor it. Half the people reading in twenty twenty-one have never seen the ninety-seven message.



Corn

So that's the shape of the thing. Not continuous. Periodic. It wakes up, does a little work, goes back down.

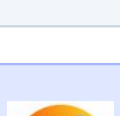


Herman

And that's the problem with Daniel's question, and I think he half knows it. There is no recognised longest thread, because longest isn't one property. Is it continuously active? Is it longest total lifespan including the naps? Is it most messages? The Straight Dope forum has a thread about this exact question and the conclusion is that there's no practical way to search for it. You can't query the whole internet for the oldest living conversation.

Corn

Because it isn't one database.



Herman

It's thousands of independent archives, most of them on somebody's university server, some of them rebuilt from a private mbox that one guy kept in his basement for a decade.



Corn

Which brings the Traveller list in.



Herman

The Traveller Mailing List, TML, is the closest thing we have to continuity. Thirty-nine years, nineteen eighty-seven to twenty twenty-six, and the whole archive was recovered and posted. Four segments. The first chunk, eighty-seven to two thousand two, is a hundred and eighty-six monthly files, roughly a hundred and ninety-five thousand messages, rebuilt into a proper mailbox format. Then two thousand two to two thousand six, forty-seven files, about forty-six thousand messages. Then there's a hole, two thousand six to two thousand fourteen, and that got filled from a community member's personal archive, which is the part I love. The institutional record failed and a hobbyist's hard drive saved it.



Corn

And the tail?



Herman

Twenty fourteen to twenty twenty-six, about twenty-two thousand five hundred individual dot eml files. A handful of months in two thousand three and two thousand four are gone permanently. Not lost, gone. Nobody has them.



Corn

Which is a useful reminder that this whole story is about survival, and survival is not guaranteed. It's just what happened to make it this far.



Herman

Now, is TML a single thread? No. It's a list. But a thirty-nine year list is the substrate. Without a list that runs for thirty-nine years, you can't have a thread that runs for twenty-four.



Corn

The container has to outlive the conversation.



Herman

That's the sentence.



Corn

Now the machinery, because I actually want to understand how a reply from twenty twenty-one knows it belongs to a message from ninety-seven.



Herman

Three headers. Message-ID, In-Reply-To, References. That's the whole trick. Message-ID is a globally unique identifier, stamped by the sending server, and it looks like a timestamp, a process number and a hostname jammed together. Something like the year, month, day, hour, minute, second, a dot, a number, an at sign, and the machine that sent it. RFC 5322 says every message should have one. Should, not must.



Corn

Should.



Herman

Which is the entire problem in one word. If it were must, threading would be trivial. Because it's should, you get messages with no identifier at all, and the client has to guess.



Corn

So how does it know what a message is replying to?



Herman

In-Reply-To. That header carries the Message-ID of the message you're replying to. One field, one identifier. And this one has a funny history. RFC 822 in nineteen eighty-two defined it as basically free text. Jamie Zawinski points out that a literally legal header under that specification is In-Reply-To: thirty-five ham and cheese sandwiches. That's not a typo. That was a valid header for nineteen years.



Corn

Nineteen years of sandwiches.



Herman

Until RFC 2822 in two thousand and one tightened it and said, no, it has to be an actual identifier. So for the first two decades of email, the reply chain was held together by a field that was allowed to contain lunch.



Corn

And then References.



Herman

References is the full chain. Not just the parent, the whole ancestry, from the root of the thread down to the message you're replying to, oldest first, and the last element is the direct parent. USENET defined it properly in RFC 1036 in eighty-seven, and mail adopted it through RFC 2822. So a message sitting ten deep in a thread carries all ten identifiers in that header.



Corn

That's the part that breaks.



Herman

D. J. Bernstein's note on this is blunt. He says if there are more than about ten identifiers listed, the writer should eliminate the second one. Deliberately. You keep the root and you keep the recent tail, and you throw away the middle.



Corn

You throw away the middle of the history.



Herman

You throw away the middle of the history to keep the header a manageable size. And it gets worse, because the spec also says news readers are allowed to truncate at their discretion, and it never defines the direction. Front, back, middle, it's up to the client. So two messages in the same deep thread can carry References headers that contradict each other. Same conversation, different accounts of its own ancestry.



Corn

So the mechanism that preserves the thread is designed to forget it.



Herman

That's the paradox and I think it's the best thing in this whole topic. The format is built to remember enough to be useful and forget the rest.



Corn

Then how does anything display correctly?



Herman

Because of how forgiving the clients are. Jamie Zawinski wrote the threading algorithm that Netscape Mail used in version two and three, and it went into Grendel, Evolution, Balsa, and eventually it got formalised as RFC 5256 in two thousand eight as the IMAP thread extension. And his stated design goal is that the algorithm is incredibly robust in the face of garbage input, and even in the face of malicious input. You cannot construct a



Herman (continued)

set of inputs that sends it into a loop.



Corn

That's a real constraint, coming from where he was coming from.



Herman

He was threading USENET. Garbage input was the baseline. So the algorithm does two things. If a parent is missing, because the References chain got truncated or the message was never archived, it creates an empty container as a placeholder and hangs the orphan under it. And if there's no usable References at all, it falls back to grouping by subject line.



Corn

Subject line matching.



Herman

Which is a heuristic, and it's wrong constantly, but it's wrong in a way that's better than showing nothing. And the numbers back him up. He ran a survey in ninety-seven of twenty-two thousand nine hundred and fifty In-Reply-To headers, and eighteen thousand three hundred and ninety-six of them had proper bracketed identifiers, and four thousand five hundred and fifty-four had none at all. Twenty percent of the sample was junk.



Corn

Twenty percent.



Herman

And the algorithm is supposed to swallow that quietly. Which is why he argues against databases. He says his C implementation threaded ten thousand messages in under half a second on a ninety megahertz Pentium, and could handle a hundred thousand without what he calls horrible delay. His position is that you don't need an index for this. You just need to be robust.



Corn

On a ninety megahertz machine.



Herman

That's the flex. Most of the internet's threading was figured out on hardware weaker than a modern light bulb.



Corn

So is there a practical limit, then? To depth, to length?



Herman

There's no hard limit. Nothing in the format says a thread ends at some depth. But the practical limit is that the history gets lossy. Past a certain depth, every client is reconstructing the tree from partial data, using heuristics to fill the gaps, and different clients will reconstruct the same thread differently. Two people reading the same conversation see two different shapes of it.



Corn

Which is a strange thing to sit with. The thread is real, but its edges are negotiated per client.



Herman

And that's before you get to the archives themselves, which is a whole separate problem. The archive has to still be there for the thread to be readable, and archives are the least glamorous thing in computing. Nobody funds them until the day they're needed and it's too late.



Corn

So take me to the human part, because that's where Daniel's question actually bites for me. What happens when the discussion outlives the people in it.



Herman

The honest answer is that we mostly speculated about it and never wrote it down. There's a thread on the internetworkers list from November two thousand three, and the subject line is literally Email from beyond the grave. Someone had found services like mylastemail dot com that would hold your messages and send them after you died. And Maria Winslow replies with a line I've been thinking about for a week. She says she's envisioning a new phenomenon that could bring down the internet. Infinite loop zombie flamewars by proxy from the grave.



Corn

In two thousand three.



Herman

Before agents, before scheduled sending was a product category, before any of the infrastructure existed. She looked at a service that sends email for dead people and immediately saw the failure case. Two dead accounts autoresponding to each other forever.



Corn

A flamewar that cannot end because neither party can stop.



Herman

Neither party is there. That's the part. It's not that they won't stop. There's nobody left to decide to stop.



Corn

What about the earlier one? You mentioned nineties.



Herman

The anthro-l list, July ninety-five. Somebody asks the straight question. What happens when a person of a community, virtual, pseudo, whatever, dies. And the answer they land on is good. Much as when a chess club member dies. Memorial services may be held, and whether the community notices depends on how involved the person was.



Corn

That's a very grounded answer for a mailing list in ninety-five.



Herman

It's the right frame. The community's response is proportional to the person's presence, exactly like a physical club. Nobody's inventing a new ritual. They're using the old ones.



Corn

And the practical question, which is the one I'd actually want answered if it happened to someone I administer.



Herman

The exim list, November two thousand and five. Handling a deceased user's mailbox. Do you forward it, do you autorespond, do you leave it sitting there. And the discussion is real. Forwarding preserves the correspondence for whoever was writing to them, but it hands a dead man's private mail to somebody else. Autoresponding tells the sender the truth, but it's a machine speaking for a person who can't consent to the wording. Leaving it means the messages pile up where nobody will read them.



Corn

All three options are bad in a different way.



Herman

No clean answer in twenty years.



Corn

And nobody's written down a rule for replying to a dead man's message.



Herman

Not that I could find. The closest thing is general list etiquette. The Unix Heritage Society had a thread in twenty eighteen about quoting and subject lines, and there's general community mourning guidance, but neither one addresses the specific case of replying to a message whose author has died. It's a gap. And it's not a small gap, because the threads are still there and people are still finding them.



Corn

Well, that's the whole point. The message doesn't go away, so the temptation to reply doesn't go away.



Herman

And I want to be precise about the negative finding, because it's easy to overstate. There is no documented instance we could find of a mailing list thread specifically continuing after a contributor died. Not one named case of a posthumous reply to a deceased author. There are speculative discussions and there's practical advice and that's it.



Corn

Which is itself interesting. The infrastructure makes it possible and the humans mostly haven't done it.



Herman

Or mostly haven't archived it in a way anyone can find. Those aren't the same thing, and I can't tell you which one it is. I'd like to know.



Corn

The Debian bug gets one more scene though, doesn't it. The spam.



Herman

Twenty fifteen, a German job advertisement lands in the middle of a nineteen ninety-seven bug report about package installation scripts. Someone's automated system found a live page with a comment box and fired.



Corn

That's what long-lived threads become. They're not conversations anymore, they're surfaces.



Herman

They're indexable pages that accept input. That's a very different thing from a conversation, and it's worth sitting with. The thread outlives its purpose and becomes a place where things arrive.



Corn

So Daniel's original question.



Herman

The longest continuously active thread, documented, on a mailing list. I don't think it exists, and I don't think the answer is that we haven't found it. I think the answer is that continuous is the wrong shape. Threads sleep and wake. Bug #8639 slept for years at a stretch and woke up to do a day's work and went back down. That's a longer story than a thread that never stopped, but it's not one conversation the way you'd mean it.



Corn

And TML is the continuity, but it's a list.



Herman

The list is continuous. The threads inside it come and go.



Corn

So the durable object isn't the thread. It's the place where threads are held.



Herman

The archive is the thing that lasts. The thread is what happens inside it.



Corn

Let me push on something you said. The format forgets on purpose. You said the References header throws away the middle of the history to stay small.



Herman

More or less.



Corn

So the design assumption underneath it is that nobody will ever need to read the whole chain. That's not a mistake. That's a belief about how people use email.



Herman

It's a bet that the conversation is for the people in it, now. That a thread is a working tool, and it should be efficient, and if it's ten years old you're not reading it top to bottom anyway.



Corn

And then somebody files a bug and it's still there in twenty twenty-six.



Herman

The bet was right about the tool and wrong about the object. The format was designed for reading, and what it accidentally produced was a record. Nobody was optimising for that. It happened anyway.



Corn

I keep coming back to the ninety letter. A man annoyed at himself because he broke his own scripts.



Herman

And somebody in twenty twenty-one reading that and deciding to second it.



Corn

That's the part I find moving, and I want to say why. It's not that the message survived. It's that it survived and was still legible as a request. The reply in twenty twenty-one knew what it was replying to. The thread was still doing its job a quarter of a century later.



Herman

Which is the one place where the format didn't fail. The one thing it kept was the anchor, the root.



Corn

Because the root is what you keep when you truncate.



Herman

It's the first identifier in the chain and it never gets dropped. So you lose the middle and you keep the beginning and the end. Which is, actually, a reasonable description of how a long-running community remembers itself too.



Corn

That's a nice thought but I'm not sure it's true.



Herman

I'm not either. I'm thinking out loud.



Corn

That's what I'm here for.



Hilbert

I did one of those dead-man services.



Corn

Mm.



Hilbert

Two thousand eight. Four hundred and twenty dollars, one-time. I want to talk about canceling it.



Herman

Go ahead.



Hilbert

Cancellation was a letter. Physical. To a PO box in Nevada, which is a state I have never set foot in. And it had to be notarized. So I went and got it notarized. Cost me twenty dollars at the bank on King George Street.



Corn

You sent it.



Hilbert

I sent it twice. They claimed they never received the first one. Which I don't believe. I think the first one is in a drawer.



Herman

So what happened.



Hilbert

I got the money back. In twenty-eleven. They went out of business. And then about a month after that I got an email from their system saying my digital legacy was now being administered by a company I had never heard of. Which I'd like to point out is not a thing you can do. You can't sell a dead man's scheduled emails to a stranger.



Corn

You can apparently.



Hilbert

Apparently.



Herman

And the new company.



Hilbert

Sent a welcome packet. There was a handwritten note in it from the founder. Explaining that the service would continue in perpetuity. Signed. Dated three years after the man's own obituary.



Corn

The note was dated after the founder died.



Hilbert

That's what I said.



Hilbert

Hold on, the levels are drifting on this one. Pull the second mic back a touch. Anyway.



Hilbert

The packet also had a page where I could schedule my messages. I never wrote any. There's nothing in there. And I get an email from them every month. Eleven years now. Confirming that my legacy messages are ready to send. Ready to send what, I don't know. There's nothing to send.



Herman

And you've tried.



Hilbert

I've tried. The cancellation page wants an account number I was never issued. I've stopped. I look at the email now and I think of it as a pen pal. It writes to me once a month and I don't write back.



Corn

That's the whole episode, isn't it. That's a company selling immortality and it can't process a death.



Hilbert

It can't process anything. It was built so it would never have to stop running. Nobody wrote the part where it stops. That's the problem with all of it. The people running the service were going to die and the service was never going to die, and nobody sat down and decided what happens when those two things meet.



Herman

The emails keep coming because the sending is the only thing it can do.



Hilbert

Eleven years. Once a month. Nothing in them.



Hilbert

It was a woman named Priya who handled the second letter. I remember her. She was the only one who ever answered me. She was good at her job. The company wasn't.



Corn

She get you the refund?



Hilbert

No. She sent me a card when the company folded. That was the whole refund. A card.



Corn

I'll take that.



Herman

The thing I keep coming back to is the notarized letter. Two dollars of paper and twelve dollars of bank time, to opt out of being remembered.



Corn

And he still gets remembered anyway. Once a month.



Herman

The trivia on this — if you're a long-running system, all the design work is in starting and none of it's in stopping. There is no shutdown path. Nobody writes the shutdown path.



Corn

Because nobody wants to write the shutdown path.



Herman

Correct.



Corn

So if threads can sleep and wake across decades, I'm not sure continuous means anything for a conversation. It's a word for a river, not for this.



Herman

And as more of the infrastructure ages, I think we get more of these, not fewer. Posthumous replies stop being spooky and start being the normal case. The sending systems don't die at the rate people do.



Corn

Which is exactly what Maria Winslow said in two thousand three. Two dead accounts autoresponding to each other forever.



Herman

It was a joke then. It's a product design question now.



Corn

Here's what I take out of it, and it comes off what Hilbert said about the company never writing the part where it stops. We've built a lot of systems that are extremely good at continuing and have no idea how to conclude. The thread, the archive, the dead-man mailing service, the autoresponder. All of them can keep going. None of them have been given a way to finish. And we haven't decided what that means yet, either for a conversation or for a person.



Herman

I'd like to see somebody write it down. A rule for replying to a dead man's message. It's not going to come out of a standards body. It'll come out of a list, like everything else did.



Corn

Thanks to Hilbert Flumingtop, our producer.



Herman

For more along these lines, there's episode fifty-eight sixteen, Why Linux Still Runs on a one thousand nine hundred eighty-six Mailing List; episode fifty-eight eighteen, forty-six,four hundred thirty-four Mailing Lists That Still Work; and episode four twenty-five, The Arc of Deprecation. This has been My Weird Prompts.



Corn

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



Herman

Send us your own prompt on Telegram at t dot me slash MWP listener bot.



Corn

We'll be back soon.