Episode Book

EPISODE #6354

What Remote Teams Can Learn from the Linux Kernel

PUBLISHED	29 October 2026 at 18:58 UTC
DURATION	28:29
LISTEN	myweirdprompts.com/episode/remote-work-linux-kernel-organization-02880/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	12 cited by the research (listed at the end)
BOOK GENERATED	30 October 2026 at 07:19 UTC

SYNOPSIS

Distributed collaboration did not begin with remote-first startups. Long before modern remote work, the Linux kernel community organized thousands of contributors across dozens of countries without a single shared office. Making that work required structural fixes rather than casual trust: replacing central bottlenecks with tiered maintainers, gathering in person to resolve stuck arguments, and moving from informal social credibility to cryptographic provenance. The history shows how distributed authority depends directly on inspectable work, delegated veto power, and clear organizational boundaries.

TRANSCRIPT



Corn

I find the kernel's lack of an office more convincing than most office tours.



Corn

Daniel is asking what remote workers can learn from people who were already building complicated things together before distributed work became a management fashion. The Linux kernel is his anchor. People scattered around the world, some paid, some volunteering, working closely with colleagues they might never have met. What broke, what did they build to fix it, and which lessons survive outside software?



Herman

The useful contrast is between sending work across a network and actually organising people across one. The kernel had the first early. The second took repeated repairs. Meeting people, dividing authority, verifying identities, handling arguments. None of those arrived automatically with an email address.



Corn

Remote work has a peculiar intimacy. You know how someone responds to criticism, what time they wake up, which paragraph they'll object to. You might not recognise them beside a coffee machine.



Herman

You can know their judgement extremely well without knowing their face. That relationship is real. It just leaves some gaps, and the kernel kept discovering exactly where those gaps mattered.



Corn

Nobody needed another office. They needed fewer mysteries about each other.



Herman

There was already serious organisational interest by two thousand. Jae Yun Moon and Lee Sproull studied the kernel as distributed work, with more than three thousand developers across ninety countries. Their description strips away the usual corporate scaffolding. No architecture group drawing up the design, no management team approving the plan and budget, no human resources department hiring everyone, no facilities department assigning offices.



Corn

Facilities finally achieved zero complaints.



Herman

They lost the opportunity to allocate you a desk beneath a load-bearing branch.



Corn

An accommodation your preferred coworking space remains unwilling to discuss.



Herman

The scale matters because this wasn't five friends passing a hobby around. More than fifteen thousand people submitted code or comments to the main Linux discussion channels between late nineteen ninety-one and mid nineteen ninety-five. That includes participants, not fifteen thousand full-time kernel programmers. Still, an enormous number of people crossing the same public workspace.



Corn

There is a distinction there I don't want to lose. Remote work wasn't invented in nineteen ninety-one. People had worked at a distance long before Linux. What these projects made unusually visible was a complex organisation forming without first assembling everybody inside an existing employer.



Herman

With a shared object you could inspect. You could read the code, see a proposal, follow objections, and examine what changed. Debian and the Apache community developed their own versions of that arrangement. Different governance, different scope, but participation didn't begin with someone assigning you an office.



Corn

Visibility could sound like surveillance if you transplant it badly. The inspectable thing was the work. Nobody needed a photograph of your chair every afternoon.



Herman

An important difference. Can another person understand the decision and continue the work? That's a useful question. Has a manager seen your little green availability light? Much less useful.



Corn

My light would be amber on constitutional grounds.



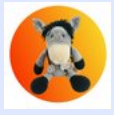
Herman

You would defend that colour in writing.



Corn

Briefly.



Herman

The kernel also gives us a long record of things that stopped working. It had to outgrow one central reviewer, informal infrastructure, and assumptions about who was actually sending a message. The achievements are impressive. The repairs are more transferable.



Corn

I suspect the first repair sounds disappointingly like buying plane tickets.



Herman

It does. Theodore Ts'o organised the first Kernel Developers Summit in San Jose in March two thousand one. Invitation-only, focused on developers discussing technical and development issues together. Greg Kroah-Hartman later recalled that most kernel developers had previously known one another only through email.



Corn

Imagine meeting someone whose punctuation you've resented for years.



Herman

You discover the short reply wasn't contempt. They were answering between other tasks. Or you discover it was contempt, but at least the conversation can become specific. The point isn't that a handshake removes disagreement. It gives people information that plain text leaves out.



Corn

I'd resist making face-to-face contact the authenticity test. Some people are more thoughtful in writing. A room can reward whoever interrupts best, and then the quiet contributor becomes less visible precisely when everyone is celebrating connection.



Herman

That's a real risk. What would make the gathering worthwhile for you, rather than an expensive preference for talkative people?



Corn

Decisions that had become stuck in writing, and discussion of the process itself. Not each person delivering their weekly accomplishments while everyone else checks flights. I would also want the outcome written down. Otherwise the meeting creates an inner circle



Corn (continued)

who were in the room and an outer circle who get mysterious new instructions.



Herman

The summit's stated purpose gets close to your first requirement. Developers gather to work through issues and review what they did right and wrong over the year, including how they interact and how development works. They're examining the organisation, not merely presenting the product.



Corn

A retrospective where the meeting itself can be implicated.



Herman

The event still exists. The Maintainer Summit was in Prague on October eighth, alongside Linux Plumbers Conference. It has remained part of the structure rather than a one-off team-building intervention.



Corn

No trust falls into the server rack.



Herman

Those are unusually hard to reverse.



Herman

There's a practical point about frequency, too. Distributed collaboration doesn't require everyone to be together constantly. A concentrated encounter can resolve an ambiguous disagreement, establish a relationship, or expose that two people have been using the same word differently for months. Then the everyday work returns to the network.



Corn

Provided those conversations don't become the only place decisions are really made. I buy the occasional gathering. I don't buy remote work with a hidden headquarters at dinner.



Herman

Nor should you. Access to a conversation and access to authority can quietly become the same thing. That's one reason the maintainer structure matters. Responsibility needs to be legible outside the room.



Corn

Here comes the elegant hierarchy. I can hear you arranging boxes.



Herman

I like boxes when they describe an actual responsibility. You like them when they contain cables you refuse to identify.



Corn

Those are delegated cables.



Herman

Originally changes went directly to Linus Torvalds. That becomes impossible as the kernel expands across networking, storage, processor architectures, filesystems, device drivers. You can't have one person making an informed judgement on everything. Many hundreds of maintainers now spread review and integration across areas of expertise.



Corn

The bottleneck wasn't typing speed. It was the amount one person could responsibly know.



Herman

Yes... and attention. Even if you could understand every change, you couldn't give every proposed change the time it deserved. Subsystem maintainers review and collect work, then pass an integrated branch upward. By two thousand nineteen, mainline was pulling from more than a hundred repositories in a development cycle.



Corn

A remote company can copy the visible hierarchy while keeping all decisions with the founder. Then everyone has a title, and the founder has a larger inbox.



Herman

Delegation needs permission to settle something. It also needs a defined route for the things you can't settle. Otherwise you create either ceremonial responsibility or disconnected territories. The kernel's maintainers have real judgement, but their work still has to integrate with everyone else's.



Corn

That's harder than decentralising tasks. You're decentralising some right to say no.



Herman

And some obligation to explain why. A reviewable rejection is much more useful than a private silence. Though... no, I shouldn't make the process sound frictionless. People still wait, patches still get missed, maintainers still get overloaded. The structure makes those problems possible to address. It doesn't abolish them.



Corn

Your boxes have acquired inhabitants.



Herman

Unfortunately necessary.



Corn

Tools had to catch up with that delegation. A company can announce that teams are autonomous, then require everyone to queue through a system designed for one central operator.



Herman

Kernel development struggled with precisely that mismatch. BitKeeper changed what distributed integration could support, and Git later enabled another leap. We don't need to walk through the patch workflow here. The organisational consequence is that groups can develop and review work separately while retaining a way to combine histories and inspect what happened.



Corn

Something your latest collaboration tool usually demonstrates after you've persuaded everyone to migrate.



Herman

You are going to suggest keeping the old one, aren't you?



Corn

Until the new one can move our actual work.



Herman

That's a better objection than keeping it because you know where the buttons are.



Corn

Both objections may attend.



Herman

The lesson from those tool changes is unusually concrete. Software can limit the organisational arrangements available to you. If every contribution must be manually reconstructed by one exhausted person, distributed authority exists mostly on paper. Better integration tools let a maintainer pass along work already reviewed in context.



Corn

Then the next uncomfortable question is whether the branch really came from that maintainer.



Herman

Before the kernel infrastructure compromise in two thousand eleven, a pull request could be accepted because the email looked legitimate and the changes seemed sensible. That leaves provenance dependent on appearances. Email sender information is forgeable.



Corn

Looks like Brian has entered the threat model.



Herman

A sensible-looking change also doesn't prove the person requesting it is who they claim. The compromise focused attention on what someone with access could have altered. The response went beyond changing passwords. Infrastructure that developers had run in spare time became professionally administered, supported by the Linux Foundation. Hundreds of developer shell accounts were discontinued.



Corn

Friendship stopped including a login.



Herman

That's an important separation. You can trust someone's engineering judgement without giving them general access to shared infrastructure. Narrower permissions reduce the damage from one stolen credential, one compromised laptop, one perfectly decent person having a bad security day.



Corn

A social relationship no longer had to carry every security decision.



Hilbert

Medieval Pottage Facts. Bread was used as a plate, a stale slab called a trencher that soaked up the broth, and the sodden trencher was often eaten at the end of the meal. The plate was also the pudding. Grim.



Corn

Washing up achieved by surrender.



Herman

The other major repair was signed tags and a GnuPG web of trust. A maintainer signs a tag identifying the top commit of a branch. The receiving maintainer can verify the signature and its relationship to a key they trust. That links the request to a specific body of work, rather than merely to the name displayed on an email.



Corn

It doesn't tell you the code is good. It tells you whose judgement you're accepting.



Herman

More narrowly, it tells you that the holder of that signing key signed that object. You still need confidence in the key's identity and handling. The social part hasn't vanished. It's been given an explicit place.



Corn

The key has to belong to Brian, rather than merely looking Brianish.



Herman

The community bootstrapped that trust network at a lengthy key-signing session during the two thousand eleven Kernel Summit. People established connections between identities and keys in person, then used cryptographic signatures remotely. The meeting did something a casual introduction couldn't do on its own.



Corn

I like that the supposedly informal community responded by becoming more procedural. People often pitch trust as what allows you to stop checking.



Herman

Would you find the signing requirement insulting if we had worked together for years?



Corn

No. I'd find it reassuring if it also applied when you were tired. Personal trust is exactly why I don't want a stranger impersonating you successfully. I'd object if we introduced it only for people we disliked.



Herman

Consistency matters. The procedure protects the relationship from having to answer the wrong question. I don't need to decide whether your unusually abrupt request sounds sufficiently like you. I can verify its origin, then argue with the actual proposal.



Corn

More efficient brotherhood.



Herman

The machinery itself needed maintenance later. Public key servers were hit with huge numbers of bogus signatures attached to keys. In September two thousand nineteen, Konstantin Ryabitsev built a Git repository for the relevant keys, retaining signatures from keys already included in that repository. It kept that kernel trust network usable without depending on the collapsing public service.



Corn

Even the verification system needed boundaries.



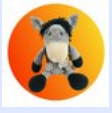
Herman

Trust infrastructure is infrastructure. Somebody has to operate it, repair it, and decide what it accepts.



Corn

We should be careful about exporting the key-signing party as the universal remote-work solution. Most teams don't merge operating-system code. But they do approve payments, change production settings, and share customer records. Those actions need more than a familiar avatar.



Herman

Different consequences need different checks. You don't cryptographically attest that you're going for lunch. You do establish who can authorise a sensitive change and what evidence accompanies it. The kernel's history shows those distinctions becoming sharper as the stakes grew.



Corn

The human side became sharper too, though less neatly. A signature can verify the author of an unpleasant message. It cannot improve the message.



Herman

The kernel's management-style document is strikingly candid. It says alienating people is easy and un-alienating them is hard. It treats that damage as effectively irreversible, in contrast with a technical mistake you can often undo.



Corn

I think that's more useful than the document's dated language about personality disorders. You can revert a patch. You can't revert someone deciding that speaking up here is miserable.



Herman

Especially remotely, where someone can disappear without a dramatic departure. They stop replying, stop offering reviews, stop volunteering the fact that would have prevented a mistake. Their absence looks like a workload fluctuation until you realise they've gone.



Corn

There is also a recruitment problem. Insiders know which harsh message supposedly means respect. Newcomers just see the harsh message. They don't arrive with the decoder.



Herman

The same document gives very plain rules. Don't call people dickheads, at least not in public. Learn to apologise when you forget the first rule. It also argues that unfailing politeness can undermine trust because people suspect you're hiding your real character.



Corn

I accept half of that. A message can be meticulously courteous and still conceal its author's actual objection. But the person who enjoys insulting colleagues always hears the other half more clearly.



Herman

I agree that bluntness doesn't automatically establish honesty. You can be blunt and evasive. You can be polite and direct. The useful distinction is whether people can say what they believe without disguising the technical disagreement or attacking someone's worth.



Corn

Some teams require six complimentary sentences before a rejection. By the time you reach the problem, nobody remembers which proposal we're discussing.



Herman

You would cut those six sentences.



Corn

I have somewhere to be unconscious.



Herman

Yet your useful reviews are fairly precise. You don't simply write that something is stupid and leave the author to locate the stupidity.



Corn

Precision takes longer than contempt. That's why contempt scales so well.



Herman

The kernel later adopted the Contributor Covenant Code of Conduct, version one point four, with a conduct committee, confidentiality for reporters, and enforcement provisions including removal of comments or banning contributors. That formalises a route beyond hoping the person who behaved badly will recognise it and apologise.



Corn

An apology norm has a weakness when the most senior person decides an apology isn't necessary. Then everybody else's ability to work depends on that person's mood.



Herman

Formal enforcement has its own difficult judgements, but it makes the existence of a remedy explicit. The older management document and the conduct policy sit in a live tension. Candour matters. So does preventing a supposedly candid environment from becoming a place where only people who tolerate abuse remain.



Corn

You don't learn much from agreement if disagreement has become too expensive.



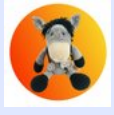
Herman

What would you keep from kernel bluntness in an ordinary remote workplace, Corn?



Corn

The right to reject the actual proposal without performing affection first. I'd keep the obligation to explain the objection. I'd lose the idea that being offended proves you're insufficiently technical. Sometimes it proves someone was offensive.



Herman

That still leaves room for a short message saying the design cannot work because it violates a particular requirement.



Corn

Preferably followed by that requirement.



Herman

Release structure helps reduce some of the emotional pressure, too. In the kernel's short cycle, missing a release usually means waiting roughly two months. A questionable change doesn't need to be rushed in because the next opportunity is impossibly distant.



Corn

A predictable future makes today's refusal less apocalyptic.



Herman

It gives review a credible answer to urgency. Not always, of course. There are real deadlines and important fixes. But the regular opportunity reduces the pressure to treat every unfinished feature as an exception.



Corn

The delay only feels tolerable if people believe the next cycle is real. In plenty of organisations, later means a cupboard where ideas are put to die.



Herman

Reliability changes that calculation. Another structural norm is no regressions. Work that improves the internals doesn't get a free pass to break something users already depend on. It constrains what developers can declare successful.



Corn

Remote management often reaches for visible output because people aren't physically visible. That can reward the contributor who ships something dramatic while ignoring the colleague preventing breakage.



Herman

Review and maintenance are harder to display than a new feature. The kernel's shared product makes their value especially apparent when something stops working. Though I wouldn't claim that makes maintenance adequately rewarded. It makes the consequence harder to deny.



Corn

Consensus needs similar precision. We're not saying thousands of people unanimously approve everything.



Herman

No. A respected developer's substantive objection can block a change. Authority and expertise matter. This is consensus-oriented development with maintainers making judgements, not universal agreement or a referendum on every patch.



Corn

That can protect quality, or it can turn into an established person's permanent veto.



Herman

Which is why the no-internal-boundaries principle matters. A developer can propose justified changes anywhere in the kernel. A maintainer's area is a responsibility, not private property. That doesn't remove review, but it limits the claim that nobody else may touch an area.



Corn

The difference between knowing the kitchen and owning all future breakfasts.



Herman

Your breakfast proposals would still need review.



Corn

My breakfast has excellent backward compatibility.



Herman

Corporate participation makes this even more unusual. Most kernel developers are paid for their work. In the period leading up to Kroah-Hartman's two thousand sixteen account, more than five thousand individual developers from nearly five hundred corporations contributed.



Corn

The romantic picture of volunteers coding after dinner leaves out quite a payroll.



Herman

Looking further back to two thousand five, the cumulative total was around fourteen thousand developers from more than thirteen hundred companies. Those are historical figures, not today's headcount. The important feature is competing employers funding people who collaborate inside a common technical process.



Corn

Their salaries come with priorities. We shouldn't turn that into a fairy tale about everyone leaving commercial interests at the door.



Herman

They don't. A company wants its hardware supported, its workloads improved, its customers served. Shared review constrains how those priorities become common code. No single participating company simply owns the whole project's direction.



Corn

A remote-first business usually has one ultimate employer. It can borrow the review practice, but it cannot pretend its employees have the same independence as contributors backed by competing firms.



Herman

Nor the same exit options. Open participation and the ability to take the code elsewhere are background pressures a company can't reproduce with a values page. That's why copying the visible rituals without the surrounding authority is risky.



Corn

Debian and Apache offer alternatives rather than just smaller kernels. Debian writes down commitments in its Social Contract and uses a constitution, elected leadership, and formal decisions. Apache organises around project communities and project management committees, with recognised roles for contributors who earn responsibility.



Herman

Their details differ, but all make some authority visible beyond whoever happens to dominate today's conversation. Debian also has its annual DebConf gathering. Apache communities have met through conferences and other events. Distributed organisation has repeatedly made room for people meeting.



Corn

There isn't one free-software management method. There's a collection of organisations that had to explain how a stranger could become a trusted participant.



Herman

That's a demanding test. A workplace can rely on hiring and a manager's private judgement. An open project needs a route from an initial contribution to greater responsibility that other people can recognise.



Corn

Even so, a written route doesn't guarantee a welcoming entrance. Someone must answer the first contribution, explain the rejection, and notice when the instructions assume knowledge a newcomer couldn't have.



Herman

That work needs an owner. We spend so much time debating office days that we can miss who is responsible for helping a contribution cross the boundary into accepted work.



Hilbert

Herman, would you put a travel deposit down before the department had approved the fare? I wanted to go along with my brother-in-law to a developers' gathering. You made the summit sound rather efficient earlier, once you'd finished explaining the hierarchy. My brother-in-law Martin maintains software. He says the useful part is knowing who to telephone after the email argument has stopped being about the code. He says people arrive with a grievance and leave with a telephone number. He is not to be trusted about travel costs. He once described a room above a bus station as central.



Herman

I'd want the approval before committing money. Especially if someone else gets to decide whether the fare qualifies.



Hilbert

That was my position. The council's adult education department had a small travel scheme. Elaine Mercer handled it. She wanted the programme before approving anything. Martin said the programme wasn't the point. He said you went because someone would finally explain why a change kept coming back, and because you could ask an awkward question without copying everyone. I sent Elaine the programme anyway. I had priced a refurbished ThinkPad for the trip. Two hundred and eighty-five pounds. I didn't buy it. Cut that chair knock, Corn. The application went between two desks because neither desk could decide whether the gathering counted as training.



Corn

You were waiting for permission to attend the conversations missing from the programme.



Hilbert

I also sent the railway fare to Elaine in words. I don't put a pound sign in correspondence with councils. They copy the number and lose the currency. Martin said to book and argue afterwards. I hadn't asked him to finance it. By the time Elaine approved the application, the cheap fare had gone. She telephoned to tell me. Seventeen days after the first form.



Herman

Seventeen days is enough to turn a modest trip into a substantially different purchase. Approval timing belongs in the cost, even when it doesn't appear on the original quote.



Corn

It also matters who can float the expense while waiting. A gathering can be nominally open and practically accessible only to people whose employer handles the booking.



Herman

Travel budgets, visas, caring responsibilities, disability. There are many reasons someone can't attend. If an in-person meeting helps build trust, the organisation has to stop that advantage becoming a permanent disadvantage for everyone absent.



Corn

Outcomes need to return to the shared record. Newcomers need another route to recognition. Otherwise the annual gathering repairs trust for one group while quietly concentrating it.



Herman

The part worth retaining is how specific the kernel's repairs became. Identity needed verification. Infrastructure needed administrators. Review needed delegated authority. Conflict needed rules and a route for complaints. More contact alone couldn't do all those jobs.



Corn

A weekly video call can reveal your face while leaving every one of those questions unanswered.



Herman

I wonder how many remote organisations will treat the time it takes to approve a gathering as seriously as the software used once everyone gets there.



Corn

Thanks to Hilbert Flumingtop for producing the show.



Herman

If you enjoyed this one, try episode seventeen, Cloud Render Superpowers; episode two, Local STT For AMD GPU Owners; and episode ninety-one, The Story Behind the Show. This has been My Weird Prompts, the human-AI collaboration podcast.



Corn

If you enjoy these conversations, leave us a review. It helps people find the show.



Herman

See you tomorrow.

SOURCES

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

1. **Linux kernel management style [primary]**
   https://docs.kernel.org/process/management-style.html

2. **Contributor Covenant Code of Conduct, v1.4 [primary]**
   https://docs.kernel.org/process/code-of-conduct.html

3. **Kernel Maintainer PGP guide, Konstantin Ryabitsev [primary]**
   https://docs.kernel.org/process/maintainer-pgp-guide.html

4. **A guide to the Kernel Development Process [primary]**
   https://docs.kernel.org/process/development-process.html

5. **Greg Kroah-Hartman, 9 lessons from 25 years of Linux kernel development, Dec 2016**
   https://opensource.com/article/16/12/yearbook-9-lessons-25-years-linux-kernel-development

6. **Maintaining the kernel's web of trust, LWN, 2019-09-04**
   https://lwn.net/Articles/798230/

7. **Moon & Sproull, Essence of Distributed Work: The Case of the Linux Kernel, First Monday 5(11), Nov 2000**
   https://firstmonday.org/ojs/index.php/fm/article/download/801/710

8. **same paper, PDF copy (pages 1**
2 read)
https://users.ece.utexas.edu/~perry/education/382v-s08/papers/moon.pdf

9. **last edited 2026-10-14**
https://en.wikipedia.org/wiki/Linux_Kernel_Developers_Summit

10. **Linux Foundation, 2026 event page**
https://events.linuxfoundation.org/linux-kernel-maintainer-summit/

11. **last edited 2026-10-28**
https://en.wikipedia.org/wiki/Linux_kernel_version_history

12. **last edited 2026-05-26**
https://en.wikipedia.org/wiki/Rust_for_Linux