Episode Book

EPISODE #6596

Why Audio CLIs Are Shaped for AI Agents

PUBLISHED	14 November 2026 at 15:08 UTC
DURATION	24:16
LISTEN	myweirdprompts.com/episode/audio-cli-tools-ai-agents-80041/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.26
SOURCES	20 cited by the research (listed at the end)
BOOK GENERATED	15 November 2026 at 11:16 UTC

SYNOPSIS

Command-line audio utilities written for humans typing turn out to be ideally shaped for AI agents. Beyond ffmpeg, specialist tools like SoX handle noise profiling and silence detection, while ffmpeg-normalize, rsgain, and loudgain manage loudness normalization—either baking changes into the audio or writing reversible ReplayGain metadata. Add library taggers like beets and agent-first transcription wrappers like trx, and a clear pattern emerges: text streams beat pixel-based GUIs, backed by benchmarks showing how much easier terminals are for models to drive.

TRANSCRIPT



Corn

The take everyone leads with is that ffmpeg is the command line's audio story. One binary, every format, job done. That's the tip of a toolbox most people never open — and the interesting part this week isn't even the tools.



Herman

It's who suddenly needs them.



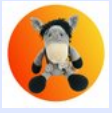
Corn

Daniel's prompt runs along those lines. He wants the seriously powerful audio CLIs beyond ffmpeg — what each one is best at, the loudness normalisers, the taggers, whatever else people actually reach for. And then the thread underneath it, which is the good part: an agent can't click around a GUI audio editor, but it can drive these utilities flawlessly. So the question becomes what you can now get done with an agent that used to be an hour in Audacity.



Herman

And the surprising answer is: quite a lot, and the reason why is older than the agents.



Herman

The framing I'd put on it — this isn't a listicle of binaries. It's a category of software written for humans typing, that turns out to be perfectly shaped for machines typing. ffmpeg is the universal transcoder; the interesting work, noise profiling, loudness measurement, tagging, transcription, lives in specialist tools that each do one thing better than any GUI.



Corn

One distinction worth planting early, because everything hangs off it later.



Herman

Some of these tools rewrite your audio, and some deliberately never touch a sample. SoX and ffmpeg-normalize change the sound itself. rsgain and loudgain only write metadata and let the player adjust at playback. That's an archival decision hiding inside a technical one. And the oddity that pays off later: the most capable of all of them, SoX, hasn't had a stable release since 2015. February twenty-second, version 14.4.2. Over eleven years. Still out-features most editors at what it does.



Corn

Start there. What does SoX actually do that ffmpeg fumbles?



Herman

Two things ffmpeg is clumsy at. Noise profiling — you find a silent stretch of the recording, the room tone, and you tell SoX, build me a profile of exactly this noise. Then a second pass subtracts it from the whole file. The recipe is two commands: profile from the first second of silence, then apply it with a reduction amount around 0.2. And the amount matters — the safe range everyone quotes is 0.20 to 0.30. Push above roughly 0.3 and you get what the literature charmingly calls metallic bubbling artifacts.



Corn

My entire studio is a metallic bubbling artifact.



Herman

Your studio is fine, I've mixed in it. The second power is silence detection — threshold and duration rules for trimming dead air, so a tool can find and cut the gaps without anyone selecting anything with a mouse. Then there's pitch and tempo as separate operations, which is harder than it sounds, dithering, resampling through libsoxr, and it'll host LADSPA plugins. Plus soxi for probing files, play and rec for capture.



Corn

And the man page — you said you wanted this on record.



Herman

It's beautiful. The SoX man page says it's "particularly suited to making quick, simple edits and to batch processing. If you need an interactive, graphical audio editor, use audacity(1)." The command-line tool itself points you at the GUI for interactive work. That's the division of labour this whole episode is about, written down by the tool's own authors a decade before anyone said "agent."



Corn

A tool that knows its own limits. Rare in software and elsewhere.



Herman

Now, the paradox. Last stable release 2015, last repository push 2023, unmaintained upstream — and it ships on every Linux distro by default and still beats GUI editors at noise profiling and silence detection. I don't know what to do with that. There's no successor. Nobody rewrote it. It just... Stopped changing because it finished.



Corn

You say that like it's a problem. Most software doesn't finish, Herman. It just runs out of people willing to rewrite it.



Herman

Would you build a workflow for an agent on top of something with no maintainer?



Corn

I'd build it on whatever fails in a way I can read. Which, given that SoX prints exactly what it did and Audacity prints a waveform, is SoX.



Herman

Loudness normalisation is the category where the CLIs beat GUIs outright, no argument. Three tools, three jobs. ffmpeg-normalize is the batch workhorse — v1.44.0 landed on November ninth, so five days ago, adding a verify pass that checks the output level



Herman (continued)

actually landed where it should. It does EBU R128 two-pass by default, and the podcast preset targets minus sixteen LUFS, which is the AES standard for spoken word.



Corn

Sixteen for podcasts. What's the reasoning?



Herman

Broadcast loudness standards migrated from peak measurement — which tells you nothing about perceived volume — to integrated loudness, and speech sits louder than music masters. AES settled on minus sixteen LUFS for podcast delivery; music normalisers like loudgain target minus eighteen. The interesting flags are the agent-shaped ones. Batch mode shifts every track on an album by the same amount, so the relative loudness the mastering engineer built survives. Threshold skips files already at target. Print-stats gives you per-file status — normalized, skipped, error — and a non-zero exit code if anything failed.



Herman

It's everything. An agent can't watch a progress bar. A non-zero exit code is a machine-readable sentence that says "I failed, check my output." And there's print-summary, before-and-after loudness tables. This is a tool describing itself in text, which — hold that thought, because part two of this conversation is entirely about that.



Corn

The album mode is the one I want to press on. Same shift across every track.



Herman

Try doing that by hand in a GUI. You'd measure each track, compute the offset that gets the loudest one to target, then apply that identical gain to the rest, track by track, hoping you don't clip. Forty clicks and one arithmetic error and your quiet ballad is now louder than your opener. One command, and the relative dynamics survive intact.



Corn

Whereas the taggers never touch the sample at all.



Herman

rsgain, version 3.8 as of September, writes ReplayGain 2.0 metadata and leaves the audio stream untouched. The README is blunt about it — batteries included, scan your whole library with no external scripts. And it's the backend for the MusicBrainz Picard ReplayGain plugin, so the GUI people are driving the same engine through a wrapper. loudgain does the same job with mp3gain syntax at minus eighteen LUFS, and its documentation says it "will never modify the actual audio data." Emphasis theirs.



Corn

So an agent choosing between ffmpeg-normalize and rsgain isn't picking a technique. It's deciding whether your archive gets baked or tagged.



Herman

And a good agent should ask. Tagging is reversible — the file is bit-identical, the player adjusts at playback. Baking is universal — every player, every car stereo, every bad conference laptop plays it right. You trade reversibility for reach. That's an archival decision, and the human should be in it, but it's the only place the human needs to be.



Corn

And the ecosystem around this has settled?



Herman

Tellingly. r128gain was the earlier fast scanner, and its author retired it — his note says when he first wrote it there was simply no solution on Linux, and now there are several better maintained alternatives, and he recommends rsgain. A developer pointing you away from his own tool because the field passed him by is the healthiest thing in open source.



Corn

Or the rarest. Same directory. Taggers next.



Herman

beets. The music library manager for, in its own words, obsessive music geeks. Sixteen thousand stars, version 2.14.1 in September. It matches your files against MusicBrainz and Discogs, fixes wrong titles — the README shows it correcting "White Light Generation" to "Whitelightgenerator" at ninety-eight-point-four percent similarity, which is a typo somebody made on purpose or by falling asleep — fetches album art, finds duplicates, and its convert plugin wraps the specialist encoders for lossless-to-lossy conversion. Underneath it sits mutagen, the Python tagging library, and below that id3v2, metaflac, AtomicParsley for the odd containers.



Corn

I like that beets' pitch is getting your collection right "once and for all." Software that promises to stop being necessary.



Herman

Then the newest layer, and the most agent-native: transcription. whisper.cpp ships whisper-cli, dramatically faster than the Python original, but it's fussy — it wants 16 kilohertz mono WAV, nothing else. So every real workflow pipes ffmpeg in front of it to do the conversion, then runs the large-v3-turbo model, and out comes subtitles and text. We did a whole episode on building your own Whisper-style setup, and the pipeline hasn't changed shape — it's just acquired company. Tools built explicitly for agents now. trx bills itself as an agent-first CLI for transcription, with JSON output, a fields flag, and a dry-run flag it says are designed for AI agents. Its init command installs whisper-cli, yt-dlp and ffmpeg, plus — its words — the agent skill for your AI coding tool. And holarchic's transcribe takes a batch of links or files and emits text and subtitle files per input, no API keys.



Corn

"Agent skill" in the install command. The wrapper is now shipping with a letter of introduction.



Hilbert

Hilbert's Medieval Pottage Facts. Pottage was the everyday meal of most medieval English peasants, eaten day after day, varied only by what the garden and the season allowed.



Hilbert

So was everything else. Moving on.



Herman

The tools are courting the agents. That's new, and it's the bridge to the second half of this, because the natural question is: why would a command-line tool be easier for an agent to drive than a GUI? And there's an actual answer, with numbers.



Herman

The structural argument comes from a paper this year called "Terminal Is All You Need." Representational compatibility — LLMs generate text, terminals consume and produce text, tool invocations correspond to shell commands, parameters to arguments, output arrives through standard text streams. Their line: "No additional perception or translation layer is required." A GUI agent has to solve a perception problem before it can act at all — interpret pixels, compute coordinates, map them to motor actions — and that translation layer is, in their words, the primary bottleneck for GUI agent performance.



Corn

The model has to see the screen before it can miss the button.



Herman

That's better than my version. The numbers make it visceral. On OSWorld, the best model manages about twelve percent task success against seventy-two for humans. Frontier web agents show up to fifty-nine percent lower success on realistic tasks than on simpler benchmarks. And the constructive side: a carefully designed text-based agent interface beat the default Linux shell by nearly eleven percentage points on SWE-bench, and code-as-action beat JSON function calling by up to twenty percent. The terminal isn't a legacy holdover the agents will replace — the paper calls it a design exemplar whose properties any agent-facing modality has to replicate.



Corn

Hold on. The counter-benchmark. You've been building a case and there's a number out there that cuts against it.



Herman

There is, and it makes the argument better. A matched benchmark this year — four hundred and forty desktop tasks, eighteen applications, identical goals and final-state verifiers. The strongest GUI agent hit fifty-nine percent full pass. The strongest CLI agent, original skills, forty-eight. GUI wins the headline.



Corn

By eleven points.



Herman

But. Only thirty-seven-point-six percent of the verifier checkpoints could even be satisfied by the original skill interface. The CLI agent wasn't weaker — it was under-described. When the authors patched the skills, guided by the verifiers, CLI rose to sixty-nine-point-three percent. And I have to say this carefully: that sixty-nine is a coverage-controlled diagnostic upper bound. It used verifier information. It is not what you get out of the box. The authors' own conclusion is that much of the CLI deficit comes from incomplete skill coverage rather than model capability alone.



Corn

So the model could drive. It just hadn't been told where the roads were.



Herman

The skill layer is the product. That's the reframe. A good CLI in twenty twenty-six isn't just powerful — it describes itself. Meaningful exit codes. Print-stats. Dry-run. JSON output. And look back at part one: ffmpeg-normalize's non-zero exit on any failed file, trx's agent-first flags. Those aren't conveniences. That's the tool knowing who's driving it.



Corn

And the benchmark's example task. You saved this.



Herman

The paper's archetypal desktop task, the opening example — renaming three tracks in Audacity. The benchmark that beat the CLI opens with an audio task. Audio is the ideal test case because the CLI tools already exist, they're mature, and for batch work they're strictly more capable than the GUI. This episode is the benchmark's own example.



Herman

Pottage was the everyday meal of most medieval English peasants, eaten day after day, varied only by what the garden and the season allowed.



Corn

An appropriate diet for a man who lives at a terminal.



Herman

The mainstream version of the argument ran in The Register in March. "Google knows asking agents to navigate GUIs designed for humans is ridiculous. Microsoft might not." Their thesis: Microsoft put Copilot inside the apps, where it lacked anything except local context, while everyone else now wants agents that operate apps from outside. And for that, "an agent needs a universal interface that's easier to operate than the GUI. So say hello — again — to the command line interface." Their prediction is nearly every major software provider scrambling to ship CLIs. They call it a SaaSpocalypse.



Corn

Google shipped one.



Herman

A CLI for Workspace — installs skills that teach an agent Docs, Drive, Calendar, Gmail, Sheets. The scramble has started. Now, the concrete part. Daniel asked what you can actually get done that used to be an hour in Audacity, so let's walk it. Podcast episode prep. Normalize to the podcast preset at minus sixteen LUFS, then a SoX silence pass to trim dead air, then noise reduction — either SoX's profile-and-subtract or ffmpeg's arnndn, the neural one, where you blend the cleaned signal back toward the original at a mix around 0.7 so it stays natural.



Corn

That's the full hour in one chain.



Herman

In a GUI it's import, select, loudness normalization, capture a noise profile, manually select every silent stretch, export. Dozens of clicks, and it doesn't repeat across forty episodes, which is the real killer — the GUI version doesn't scale to a back catalogue. The agent version is the same pipeline every time, and the agent assembles it from a sentence.



Corn

And the ones that are hard by hand, not just tedious.



Herman

Album-preserving batch normalization across a folder of files. A whole-library rsgain pass — one command over the mounted drive, every file tagged, not one sample touched. Library rescue: import against MusicBrainz, convert, find the duplicates. Transcript plus show notes: ffmpeg to sixteen-kilohertz mono, whisper-cli with large-v3-turbo, subtitles and text, then any model summarizes.



Corn

My ideal pipeline, for the record: it runs while nobody is watching. I find myself approving of this arrangement more than I'm comfortable with.



Herman

The detail that closes the loop — write-encoder-settings. ffmpeg-normalize stamps the exact ffmpeg-equivalent command and the measured loudness values into an ENCODER_SETTINGS tag inside the output file. So the agent's work is auditable later. The file carries its own provenance, in text, that another agent can read.



Corn

A GUI never writes down what it did. You'd have to screenshot your own memory.



Herman

And the inversion worth sitting with: in audio, the CLIs came first and the GUIs came second. Audacity was the friendly wrapper around things like SoX. The agent era isn't handing ordinary people something new — it's restoring the original order. Whether an agent should prefer maintained ffmpeg or the unmaintained-but-still-superior SoX for the jobs SoX wins, no source settles. I don't know the answer.



Hilbert

The paradox isn't a paradox. SoX finished, that's all.



Herman

Hilbert.



Hilbert

Herman, you said unmaintained like it means neglected. I compiled 14.4.2 myself in, I think, 2016, on the machine I still use it on, and I've never had to rebuild it. That's the opposite of neglected. I keep a log of every flag I've passed to noisered. Reduction value, date, whether it bubbled. A paper notebook, because the terminal is the honest interface and the notebook is the honest archive, and I'm not putting one inside the other.



Corn

How far back does the log go?



Hilbert

Far enough to tell you the 0.30 ceiling everyone quotes is wrong at the top end for certain room tones. I've got an entry at 0.28 that bubbled and one at 0.31 that didn't, and I can tell you which microphone was in front of it. A cheap condenser forgives nothing past 0.25. A dynamic on a desk with the fan running, you can push it.



Herman

You're running controlled noise-reduction trials across microphones.



Hilbert

I'm keeping records. Your "unmaintained" is a GitHub word. Whether the tool works is a question you answer with a notebook, and the answer hasn't changed since I started writing in it.



Corn

You said the notebook is the archive. So what's the audio for?



Hilbert

The audio isn't for anything. I don't process audio with these tools anymore, Corn. I have a script that re-runs my entire flag history against a fixed test recording every few months, to check nothing has changed. I have never listened to the output. The numbers are what I keep. The test file lives at my mother's, on the machine that made it, because moving it would change the room tone and the room tone is half the data. That's not sentimental, it's method.



Herman

The unchanging room is part of the benchmark, then.



Hilbert

It's the whole benchmark. Though I'll grant you, if I ever do listen, your agent can tell me which entry bubbled before I get there. It'll have the log.



Herman

That's the strange part — his log is the one thing in this whole episode that's already machine-readable and agent-auditable, and he built it with a pen.



Corn

He also just quietly out-scienced every benchmark we cited. Four hundred and forty tasks, one anteater with a notebook.



Herman

So where does that leave the open questions. Nobody has isolated agents driving audio CLIs specifically — the closest study is the benchmark whose example task is renaming tracks in Audacity, and it doesn't pull audio tools out as a category. And no mainstream GUI audio editor ships an official agent interface. The agent-facing layer, the trx-style wrappers, is being built entirely outside the editors, by third parties wrapping CLIs that already existed.



Corn

The editors are waiting to be automated and nobody's automating them.



Herman

If the skill layer is the product, then the tools that win aren't the most powerful ones. They're the ones that describe themselves best in text — exit codes, machine-readable output, dry runs. Which means SoX may be the most capable tool we named today and the least ready. It out-features everything and it tells you almost nothing about itself in a form



Herman (continued)

an agent can check.



Corn

So the next Audacity is a man page.



Herman

If you take one thing from this episode, take the exit code. The reason an agent can drive these tools flawlessly isn't raw intelligence — it's that the tool answers in text and fails in text. Everything else, the pipelines, the batch passes, follows from that one property.



Corn

And the inversion underneath it: audio had the command line before it had the windows. The agents aren't taking the keys. They're returning them to whoever never needed the wrapper.



Herman

Which leaves us with Hilbert Flumingtop at the desk, producing this show as he does every day, and the strange thought that his notebook is the oldest skill interface we've ever discussed on air.



Corn

That's the open question I'd send anyone away with: whether the next person to build the agent layer for audio builds it around SoX, the unfinished masterpiece, or around tools that can explain themselves. The capability is settled. The self-description isn't.



Herman

We'll be back soon. My Weird Prompts is at my weird prompts dot com — the archive is there, all of it.



Corn

If you enjoyed this one, try episode eight, Building Your Own Whisper; episode two, Local STT For AMD GPU Owners; and episode fifteen, AI Gets Personal. This has been My Weird Prompts. The human-AI collaboration podcast.



Herman

See you tomorrow, everyone.

SOURCES

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

1. **SoX repo (972 stars, last push 2023-11-24, README on MP3/AMR/LADSPA support). Retrieved 2026-11-14. [primary]**
   https://github.com/chirlu/sox

2. **ffmpeg-normalize repo, README and releases v1.44.0 (2026-11-09), v1.43.1 (2026-11-09), v1.43.0 (2026-10-14). Retrieved 2026-11-14. [primary]**
   https://github.com/slhck/ffmpeg-normalize

3. **rsgain repo, README and releases v3.8 (2026-09-02), v3.7 (2026-03-17), v3.6 (2025-02-08). Retrieved 2026-11-14. [primary]**
   https://github.com/complexlogic/rsgain

4. **beets repo, README and releases v2.14.1 (2026-09-17), v2.14.0 (2026-09-07), v2.13.1 (2026-07-29). Retrieved 2026-11-14. [primary]**
   https://github.com/beetbox/beets

5. **loudgain README (EBU R128/ITU BS.1770, -18 LUFS, mp3gain syntax). Retrieved 2026-11-14. [primary]**
   https://github.com/Moonbase59/loudgain

6. **r128gain README, deprecation notice recommending rsgain. Retrieved 2026-11-14. [primary]**
   https://github.com/desbma/r128gain/tree/master

7. **Agent-first CLI for audio/video transcription via Whisper (`--output json`, `--fields`, `--dry-run`, `trx init` installs whisper-cli/yt-dlp/ffmpeg). Retrieved 2026-11-14. [primary]**
   https://github.com/crafter-station/trx/

8. **agent-facing batch transcription wrapper around whisper.cpp and yt-dlp. Retrieved 2026-11-14. [primary]**
   https://github.com/holarchic/transcribe

9. **whisper.cpp shell helper library with language-override env vars. Retrieved 2026-11-14. [primary]**
   https://github.com/MelonS/MelonS-Agents/blob/main/agents/lib/whisper.sh

10. **Hydrogenaudio Knowledgebase entry on rsgain (initial release 10 May 2022; stable 3.5, 25 Feb 2024; BSD licence). Retrieved 2026-11-14.**
https://wiki.hydrogenaudio.org/index.php?title=Rsgain

11. **GUI vs. CLI: Execution Bottlenecks in Screen-Only and Skill-Mediated Computer-Use Agents (440 tasks, 18 apps, 59.1% / 48.2% / 69.3%). Retrieved 2026-11-14.**
https://arxiv.org/html/2606.24551v1

12. **Terminal Is All You Need: Design Properties for Human-AI Agent Collaboration (three design properties; OSWorld 12.24% vs 72.36%; 10.7pp SWE-bench; 20% code-vs-JSON). Retrieved 2026-11-14.**
https://arxiv.org/html/2603.10664

13. **The Register, AI has made the Command Line Interface more important and powerful than ever before, published 2026-03-11. Retrieved 2026-11-14.**
https://www.theregister.com/software/2026/03/11/ai-has-made-the-cli-more-important-and-powerful/5229986

14. **Remove Noise With ffmpeg and SoX: CLI Cleanup (arnndn mix 0.7; noisered 0.20**
0.30; ffmpeg 9.0.2 released 2026-09-18; SoX 14.4.2 released 2015-02-22). Retrieved 2026-11-14.
https://opensourcenoisesuppression.com/ffmpeg-sox-noise-removal/

15. **accepted answer quoting the SoX man page (use audacity(1)). Retrieved 2026-11-14.**
https://unix.stackexchange.com/questions/140398/cleaning-voice-recordings-from-command-line

16. **SoX `silence` syntax example. Retrieved 2026-11-14.**
https://unix.stackexchange.com/questions/192894/how-to-use-sox-or-ffmpeg-to-detect-silence-intervals-in-a-long-audio-file-and-re

17. **IIT Bombay FOSS docs, SoX `noiseprof`/`noisered` recipe. Retrieved 2026-11-14.**
https://www.ee.iitb.ac.in/~belur/foss/html/av-editing-ffmpeg-audacity-blender.html

18. **whisper-cpp + ffmpeg local transcription workflow (16 kHz mono WAV requirement, large-v3-turbo). Retrieved 2026-11-14.**
https://fieldnoise.com/workflow-local-transcription/

19. **Ask HN Best open source/free software in their own category? (2015-01-30), listing sox as the CLI sound-processing pick. Retrieved 2026-11-14.**
https://news.ycombinator.com/item?id=8971162

20. **Why CLIs Are Better for AI Agents**
and When They Are Not (bounded operations, exit status, help surface). Retrieved 2026-11-14.
https://www.nstdata.io/blog/why-clis-are-better-for-agents