Using AI as a Coding Assistant with Xrootd

Three small experiments to discover what an AI coding assistant gets right, gets wrong, and where a human has to step in

Xrootd & FTS Workshop, Wed, Sep. 16

Presented by Sarah Yang, Andy Hanushevsky, Wei Yang,

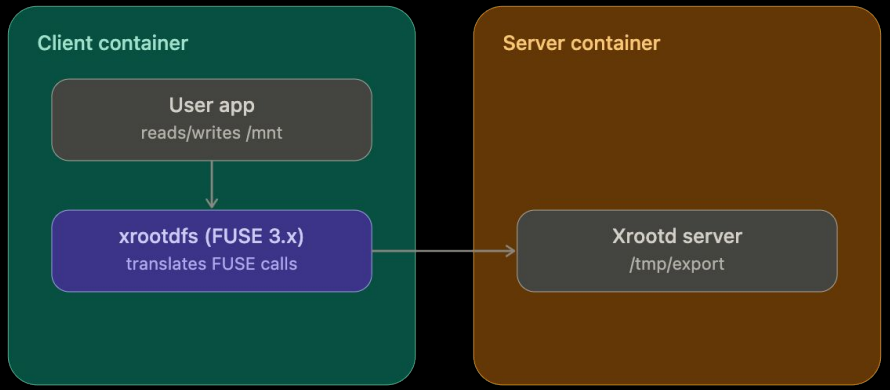
#1: Static Adler32 Tool

- Goal: standalone checksum tool where no xrootd install is required at runtime
- AI used: Claude (Anthropic) via its chat interface with code execution tools enabled
- Right:
  - Correct core design: reused xrootd's own checksum logic, static-link Xrootd libs
  - Matched official xrdadler32 behavior
- Wrong:
  - Wrong library structure assumption (missing library error)
  - No static library existed yet (had to convert XrdUtils to build both forms)
  - Missing compiler flag broke shared build
  - Underestimated dependency chain (remote root support was dropped)
- Human had to diagnose each real build error in turn, make the call to cut remote-url support, and verify final output against real Xrootd source

#2: FUSE 2.6 → 3.x Migration

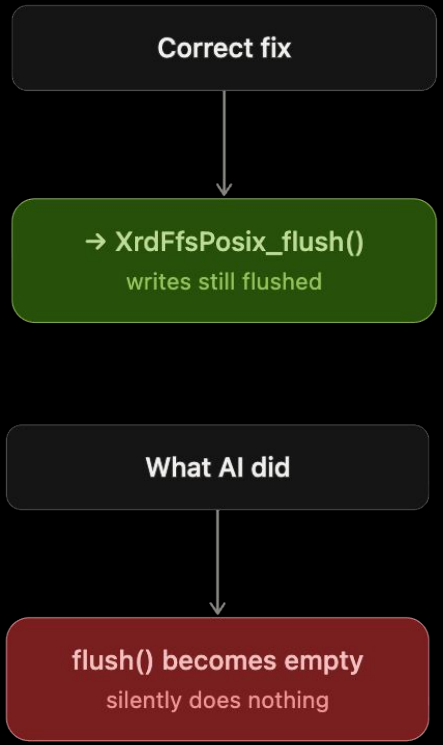
- Goal: modernize xrootdfs and prove it still works
- Right:
  - Found and fixed ~8 real FUSE 3.x API changed
  - Verified via real compile and live Docker mount test (ls,df, cp, rm all worked)
- Human had to review each change and run the actual mount test to confirm real functionality, not just compilation

purple = the code modified



#2: FUSE 2.6 → 3.x Migration

- Flush()/Wcache Trap:
  - Asked to a) add xrootdfs_flush() and b) remove the Wcache caching layer entirely
- flush()'s only job was calling into Wcache
- AI deleted that call instead of redirecting it so flush() became a no-op
  - Should've become XrdFfsPosix_flush(), not empty
- Why?
  - flush() was newly written by AI with less surrounding context than older functions
  - Other functions had a real, independent action left over after removing the cache call and flush() had nothing else so removing the one line emptied it
- Human had to catch the error, so semantic review is still essential



#3 Asking sci-token questions in root protocol

- **Trying to find out how to pass sci-token via CGI in root://... URLs**

- Asked Anthropic Opus and Google Gemini, both said using
  `root://... ?authz=Bearer%F<token_string>`
  - Opus checked the local copy of the Xrootd repo before answering
  - Google Search with Gemini made the same mistake

- Of course it didn't work – This will only work in https://... URLs
- we had to check the code in XrdSecztn to find how to do it
  `root://... ?xrd.ztn=<token_file>`

- It looks like AIs are confused about how to pass tokens via CGI in HTTP protocol and xroot protocol.
- Changed our question to : how to pass ztn token via CGI in root://... URLs
  - Opus will give the correct answer this time.

Conclusion

- I would trust AI to write code but I think there should still be human review because human prompts don't provide context so AI can very easily misunderstand the task and/or simply make mistakes
- Most efficient way to use AI would be to use AI as a first draft and then to review it by a human and like it's pair-programming during review
  - flush()/Wcache bug compiled and linked cleanly so if there was no code review we never would've known
  - Adler32 error with missing library, no static build target, a sed pattern only matching 1/12 files, missing compiler flag, underestimated dependency chain were all loud errors because compiler or linker wouldn't proceed