LLM-assisted WCT testing cleanup

Brett Viren

May 14, 2026

Overview

Issue 474 describes problems with WCT `<pkg>/test/test*.cxx` style unit tests:

- We have about 100, each has a ‘main()‘ and needs a link leading to long build times.
- Each is a isolated monolith, no way to run “suites” nor individual test cases.
- Thoughtless `cout/cerr` spray, no logging.
- Some **do not even test** (no asserts) and are more demos or rely on human examination of output.
- Some require command line args or input files.

We also have about 250 Doctest test cases which avoid most of these problems.

Want a magic converter

$$\text{test} \rightarrow \text{doctest}$$

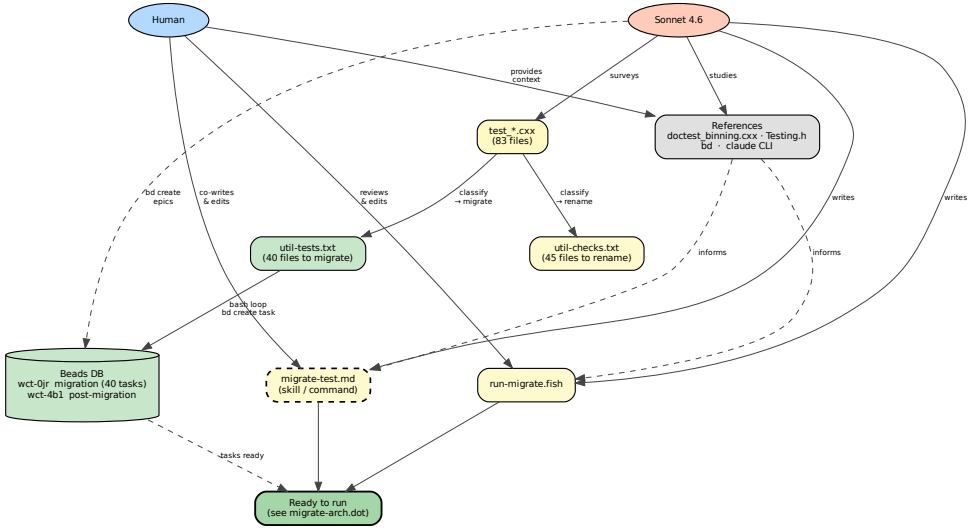
Doing this by hand is tedious.

- LLMs excel at tedium

Just do it

<user> Hey claude, convert all this crap to gold.
<claude> Uh, okay, uh,... uh... ouch, my context just exploded.

Construct a partitioning of the problem - diagram



Construct a partitioning of the problem - description

1. LLM surveys/categorizes each test: “migrate to doctest” vs “convert to a check”
   - This is quick and can be done in the main agent with a simple ad-hoc prompt.
   - Target `util/test/`, LLM makes: `util-tests.txt` listing “tests that actually test”.
     - ★ And `util-checks.txt` for tests that are better as WCT “check” programs.
2. LLM iterates `util-tests.txt`, makes a Beads issue for each test.
   - It writes a deterministic Bash loop calling `bd create` for each.
3. LLM writes a “claude command” Markdown file with arguments: bead issue + test file name.
   - It’s content described next.
4. LLM writes a shell script `run-migration.sh`
   - Calls ‘claude -p‘ on one Beads issue or can iterate on all remaining.
   - Allows a fresh LLM context for each issue and a human-friendly CLI.
   - Can load new issues and rerun script for new `test*.cxx` to migrate.
5. Human reviews the “command” Markdown file and the script.
   - Smoke test a single migration before letting the script loop over the remaining.

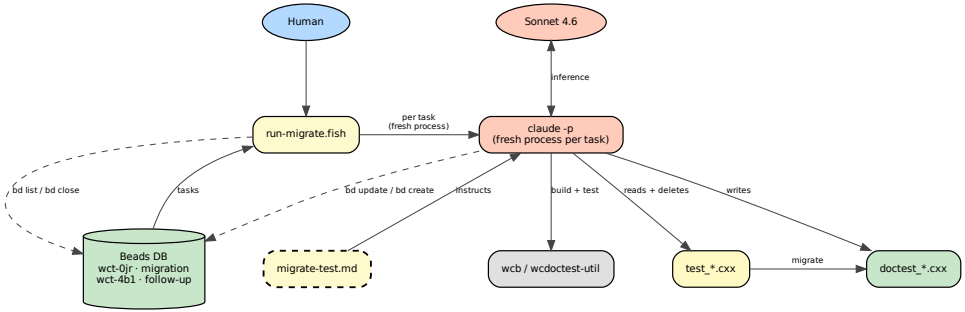
The migration "command" `migrate-test.md`

The "command"[1] is just a markdown file with $ARGUMENTS and LLM prompt content:

- Parse arguments, form Doctest file and suite name based on old `test*.cxx` file name.
- Read the `test*.cxx` source, write new `doctest-*.cxx` source:
  - Replace `cout/cerr` with `spdlog::debug()`, remove `main()`.
  - Identify test-like functions, make each into a Doctest `TEST_CASE()`.
  - Map existing types of `assert` to equivalent in Doctest.
  - Add `CHECK()` wherever it looks good but include a `CHECK(true)` to mark unchecked code.
- Post-migration actions:
  - New doctest must build, give build instruction.
  - New doctest must pass, fix test until it does.
  - Remove old `test*.cxx` file.
  - Update Beads issue with summary and close.
  - Make new Beads issue if human follow up is warranted.

[1] A Claude "command" is run explicitly while a "skill" is chosen by the LLM to run.

Exercising the workflow



- Most of the “smarts” is in `migrate-test.md` “command” to generate the prompt.
- Beads tracks progress and allows state to survive loss of an LLM session.
- The `run-migrate.sh` gives a simple high-level CLI for the human to type.

Doing the migration

Single target smoke test

```
./run-migrate wct-0jr.36   # <-- that's a beads issue ID
```

Make sure all the machinery works with a single target.

- Minor outer shell script bugs found/fixed.

Do full loop

```
./run-migrate              # no arg means loop on issues
```

Run ‘claude -p‘ with the “command” serially on each Bead issue.

- Each conversion takes about 1-5 minutes.
- Can watch progress on terminal or go get a coffee.
  - No stopping, uses `--allowedTools` to pre-authorize only tools that are needed.

Summary of resulting DocTests

40 `test*.cxx` become 91 TEST_CASE's in about 1 hour of processing time.

```
[doctest] test cases:   244 |   244 passed | 0 failed | 0 skippe
[doctest] assertions: 13685 | 13685 passed | 0 failed |
[doctest] Status: SUCCESS!
```

New issues generated

The LLM decided 3 from 40 migrations need human follow up.

- No assertions in original and none could be determined by the LLM.
- No assertions and non-trivial to invent some without deeper checking of code under test.
- Original assertion commented, human needs to define intention.

The other half of the problem

45/85 are categorized as “migrate to a check”
Roughly equal sized sub-categories (human sorting)

- Demo code just to see how some code works.
- Any real “testing” requires interpreting stdout or generated figures.
- Requires file input needing fodder and/or does file output needing tempfile support.

We can iterate this method to handle these with novel “command” files.

Summary

Translating about 40 annoying tests to 91 better ones was automated with LLM.

- Claude Sonnet 4.6.
- Took about 1 hour of setup (half day if you count constant interruption of daily stuff).
  - Would take weeks of painful hand-coding.
- Took about 1 hour of actual processing time.

Migration process had structural design driven by LLM limitations.

- Beads issues for persistent task information.
- A “command” to for consistent, parameterized instruction prompt.
- Fresh context (`claude -p`) for each file conserved token usage.
- An overall script loop for serial automation.

Caveats and comments on generalizing method

We do a really **bad job** of regularly running `test*.cxx` tests so even with **low-fidelity** results, the migration is **low risk**.

- We also do a bad job of running Doctests but that's another story.
- Higher risk code would need more human attention to the migration process and validation.

Gauged the time required by doing a few single-file migrations.

- Fast enough to let things run serially.
- Larger migrations may push to adopt some parallelism.

FIN

Backups