Improving Clang Error Recovery for Interactive Workflows

Sahil Patidar

C++Alliance Fellowship 2026
Mentors - Vassil Vassilev, Aaron Jomy

Background.

From Batch Compilation to Interactive C++

Traditional Clang was designed for **batch compilation**:

- Compile a complete translation unit
- Produce diagnostics
- Exit

Interactive environments such as Clang-Repl process code incrementally:

This changes the assumptions of the compiler:

- State persists across inputs
- Errors must be recoverable
- Each submission should behave as an isolated transaction

The Challenge

What Happens After an Error?

When an input fails, Clang may still retain parts of the semantic state:

- Declarations remain visible in lookup
- Redeclaration chains may be updated
- Template bookkeeping may be modified

As a result, later inputs can observe artifacts from failed code.

```
clang-repl> int x =
error
clang-repl> int x = 42;
error: redefinition of 'x'
```

A failed input should not affect future submissions.

Project Goal

The objective is to make failed submissions behave as though they never happened.
Key goals:

- Prevent semantic-state pollution
- Keep interactive sessions stable
- Ensure consistent recovery across Clang subsystems
- Preserve existing compiler correctness guarantees

Project Direction

Selective Rollback
Rollback only the failed submission.

Sema State Restoration
Restore compiler state to its previous snapshot.

Input Ownership Tracking
Track declarations and semantic artifacts introduced by each input.

PTU Memory Management
Enable efficient cleanup of AST/PTU allocations.

Shared Infrastructure
Provide reusable mechanisms for Clang-Repl, Cling, and future tooling.

Why it matters

Enabling Robust Interactive C++
Reliable recovery is essential for:

- Clang-Repl
- Cling
- Interactive development tools
- Future incremental compilation workflows

Without strong recovery:

- Invalid program state propagates across the session
- Subsequent compilations become unreliable
- Interactive workflows are frequently interrupted
- Long-running sessions become difficult to maintain
- Poor user experience for interactive development

🌟 Impact.

- Reliable rollback after failed inputs
- Correct semantic state restoration
- Efficient PTU-level cleanup
- Reusable infrastructure for interactive tooling
- Stronger foundation for incremental compilation in Clang

Thank You!

👨‍💻 GitHub: https://github.com/SahilPatidar/

📧 Mail: sahilpatidar60@gmail.com