Write-and-Do-Mini-Assignment#2: Error Messages in Clang, GCC and LLVM

Sagar Jain
CS17BTECH11034

September 9, 2019

Error Handling in Clang/LLVM

Programmatic Errors

The observations on error messages have been made from here. In LLVM all the errors are classified into two types; *programmatic* and *recoverable*. Assertions are used heavily in the llvm-project. An assertion can be made to check if any code invariant or condition is being broken.
For example:
In `clang/lib/Analysis/CallGraph.cpp` we can find the following:

```
assert(*CI != Root && "No one can call the root node.")
```

We know that in a cfg there can be no calls to the root node, this assert is placed in the code to ensure this condition and also holds a message so that the user can know why the program has stopped execution.

Recoverable Errors

Recoverable errors are mostly the errors which occur because of reasons other than mistakes in the source code. These errors should be reported to the user as well. Reported errors are handled using the error scheme provided by LLVM. The error class can be used for any user defined errors and we can also specify information regarding the error in it. Template functions like `make_error` are provided to construct failure values for the error class created by us which inherit from `ErrorInfo`.

The following files define the most important error handling apis and methods in LLVM:

1. `ErrorHandling.h`: This file defines an API used to indicate fatal error conditions.
2. `Error.h`: This file defines an API used to report recoverable errors.

3. `ErrorOr.h`: Provides ErrorOr⟨T⟩ smart pointer.
   This is a very important pointer. This template pointer either has the value of an operation of type T or an error, on error we not only have the error code but also other user information which further helps in debugging.

4. `Errno.h`: This file declares some portable and convenient functions to deal with errno.

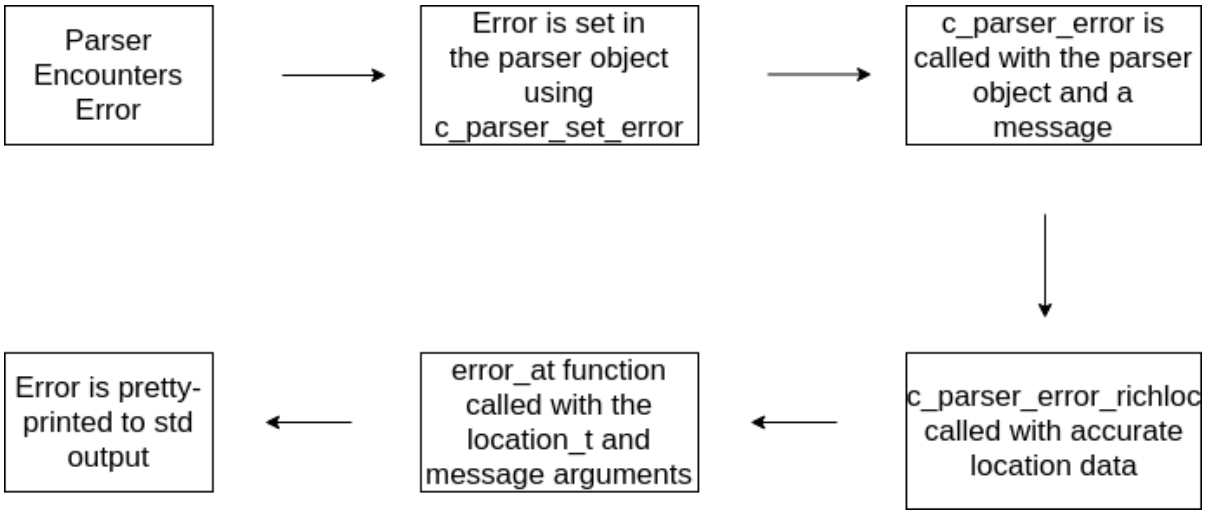
Error Handling in GCC

GCC reports two kinds of debugging information to users: *warnings* and *errors*.
For C the, file where most of the error/warning methods are defined is `gcc/diagnostic-core.h`.
Some most important functions used by gcc to report errors are:

- error_at
- warning_at
- fatal_error
- pedwarn (pedantic warnings), etc.

These functions are not called directly but through local functions defined at the level any one compiler component. Almost all the error reporting follows the same pattern where the error is discovered at the particular component and is sent down through a fixed set of functions to one of the functions mentioned above.These functions usually take arguments of the type location_t.

The following is an example of how a parser-error would make it to the user in gcc. Observations made from source files at link



Used this for flowchart construction.