GSOC 2019 Proposal
Header generation for C/C++

1. Introduction

In recent years, the D programming language has gained more and more attention and existing C and C++ codebases are starting to incrementally integrate D components.

In order to be able to use D components, a C/C++ interface to them must be provided; in C/C++, this is done through header files. Currently, this process is entirely manual, with the responsibility of writing a header file falling on shoulders of the programmer. The larger the D portion of a codebase is, the more tedious the task becomes: the best example being the DMD frontend which amounts to roughly ~310000 lines of code for which the C++ header files that are used by other backend implementations (gdc, ldc) are manually managed. This is a repetitive, time consuming, and rather boring task: this is the perfect job for a machine.

2. Project goals

The deliverable of the project is a program that automatically generates C/C++ header files from D module files. This can be achieved either by a library solution using DMD as a Library, or by adding this feature in the DMD frontend through a compiler switch.

The advantage of using DMD as a Library is that this wouldn't increase the complexity of compiler frontend codebase. The disadvantage will be that the user will be required to install a third-party tool. Contrasting to this, the addition of the feature to the frontend would result in a smoother integration with all the backends that use the DMD frontend. Choosing between the two doesn't influence the implementation in a significant way and it will be up to the language maintainers to decide.

3. Implementation strategy

The feature will require the implementation of a `Visitor` class that will traverse the `AST` resulted after the parsing phase of the D code. For each top-level `Dsymbol` (variable, function, struct, class etc.) the associated C++ correspondent will be written in the header file.

If the library solution is preferred, the implementation will use the `ParseTimeVisitor` class provided by `dmdfelib`. The `ParseTimeVisitor` will be inherited by a `CppHdrGenVisitor` that will override the visiting methods for two types of nodes:

- Traversal nodes - these nodes simply implement the `AST` traversal logic: `ModuleDeclaration`, `ScopeDeclaration`, etc.
- Output nodes - these nodes will implement the actual header generation logic: `FuncDeclaration`, `StructDeclaration`, `VarDeclaration`, etc.

An additional step to the library solution implementation will be to initialize all the DMD specific internals in a main function, similar to what `mars.d` does. This is required as DMD as a Library does not offer an easy initialization function.

If the feature is integrated in the compiler frontend, the implementation can either be integrated in the `hdrgen.d` file and simply add the `CppHdrGenVisitor` that implements the output nodes as stated above.

Since C++ is a superset of C, the implementation will be split in two phases:

- Firstly, it will implement the header generation for the C subset, as any valid C program is a valid C++ one.
- Secondly, the C++ header generation will be built upon the C subset, treating the specific cases: classes, ctors, dtors, member functions, etc.

4. Timeline

Week	**Task - Translate item**
May 27th - June 3rd	Global variables
June 3rd - June 10th	Manifest constants
June 10th - June 24th	Free functions
June 24th - July 1st	First evaluation

July 1st - July 8th	Imports
July 8th - July 22th	Struct/union types
July 22th - July 29th	Second evaluation
July 29th - August 12th	Class types
August 12th - August 26th	Explore template declarations

5. Expected outcome

At the end of the summer I expect that I will have implemented a fully operational C/C++ header generation tool. Hopefully, I will be able to integrate this with the DMD frontend codebase in order to replace the manual header generation process.

6. About me

I am a PhD student and Teaching Assistant at University "Politehnica" of Bucharest. I believe I am a hard working student who enjoys to get his hands dirty.

I like learning new technologies and strengthening my current knowledge. I am passionate about computer science, programming languages, coffee and sports. My previous experience includes distributed systems and parallel programming, operating systems, basic kernel development, open-source software and basic Android programming. I am a Linux fan, a command line addict and a vim enthusiast. I hope that through my work I will be able to help and improve the D language, which I am becoming so fond of.

6.1. Contact information

- Email: edi33416@gmail.com
- LinkedIn: https://www.linkedin.com/in/constantin-eduard-staniloiu-35266437/
- Github: https://github.com/edi33416