Overview

`parse_gen` is a sample LL(1) C parser generator. It takes 2 arguments for the source file and the output c file along with an optional 3rd argument specifying an output c header file. The source file is in a syntax similar to `bison/yacc` however there are key differences as this application shell **not** accept bison grammer files and vis versa. These differences include (*but are not exclusive to*) different prefixes for commands as well as a reduced command set. Specification of the proper input file format is documented here.

Once the output c file is generated, one may compile this file with any complying c copmiler along with any other files used in complilation. The outputted file shall include a definition for a parser taking configured arguments and running symantic actions based upon the passed grammer. This function will often be named `yyparse`, however the prefix may be modified via a command in the grammer file.

Building

Before building this project, be sure your environment has a relatively recent verision of GNU `make`, `bison`, `flex`, and a compiler with compatibility with the `gnu99` C standard and common gcc flags.

To build the `parse_gen` executable in release mode, run either `make` or `make all`. This shall build with optimization towards speed and without debug information. Should debug information be desired, run `make debug` which shall build `parse_gen` still with full optimaztoins, but this executable shall have full dwarf debugging information.

For further modification of the build process, there are the following variables which can be passed at the end of make:

- `V=1`

  This shall have the build be run verbose mode, showing the full commands being run during building.

- `CC=<name>`

  This sets the compiler used to be `<name>` for all actions during the build.

- `D=<...>`

  Adds custom flags to be passed for every compilation of a c file. For more compilcated flags or multiple flags, it is highly recommended to wrap the `<...>` by double quotes.

Examples

Example projects and code that use `parse_gen` can be found in the `example` directory. These include:

- Basic grammer files which generate code for building parse tree from a built-in token stream
- More complex programs with a `Makefile` which can be used to build them

Be sure that `parse_gen` is built in the project root directoy before attempting to compile any of the above examples.

Documentation

Full documentation for `pase_gen` can be found here.