Compiler Documentation

Thomas Maloney
net-id tmaloney@iastate.edu

February 2021

1 Part 0 Documentation

1.1 main.c

This file contains the entry point of the compiler. It passes the arguments from the command line to a method that parses them (described in section 1.2) and returns a `struct` that contains info about what flags were passed and what the possible output file might be (should that have been given). I then check which flag got passed in and whether the output should be written to a provided file or just `stdout`. The only `mode` flag that currently gets handled is `-0`.

1.2 args_parser

The header file contains a `struct`, an `enum` (both described in section 1.3), and a method signature that gets implemented in `args_parser.c`. Here I use `getopt` from `unistd.h` to read/parse each flag and then record which flag and whether there was an output file to a `struct` that I return. The potential output file gets passed back as an out-parameter. If no flags were passed in, or one that doesn't exist, I print out the string that describes how to use the program to `stderr`.

1.3 Data Structures

There is one `struct` and one `enum` that have been defined so far:

- (`struct`) `parsed_args_t`: This data type holds what `mode` flag gets passed in by way of the `mode_e enum`. It also contains a flag that signals if there is a specific output file that should be written to instead of `stdout`.
- (`enum`) `mode_e`: This just represents the possible `mode` flags that can be passed in plus an extra one that is set by default to make it easier to check if none of the required flags were passed in.

2 Part 1 Documentation

2.1 main.c

For the most part, everything here is pretty similar to what's described in subsection 1.1. That is, not much has really changed. The one thing that has, however is that it now grabs the parsed file names that get stored in a `parsed_args_t` instance after which it sets a global variable equal to the array. This allows the generated lexer file to access the list of files it needs to lex.

2.2 lexer.l

Contains the regex rules for matching tokens. There is also a function in it that allows `main.c` to set the starting input file from an `extern` string array that also gets pulled in from `main.c`. Currently the lexer will just ignore preprocessor symbols.

2.3 log_utils

Provides two function for logging data (info and errors) to the output stream, templated to match the assignment's specifications.

2.4 args_parser

Pretty similar to its implementation described in subsection 1.2. A new feature in it is the ability to parse the input files from the command line and store them in an array.

2.5 Data Structures

There is one `struct` and one `enum` that have been defined so far:

- (`struct`) `parsed_args_t`: This data type is still similar to how it is described in subsection 1.3. It has only changed in that it now keeps track of the input files.
- (`enum`) `mode_e`: Same as described in subsection 1.3.
- (`enum`) `token_e`: Enumerated list of syntax token types.