

## Blog

Written words that are often helpful

### Compiler Development 101

Posted by chathaway on Sept. 2, 2017, 9:33 p.m.

Many sites offer tutorials on how to use things like flex, bison, and LLVM. However, these sites give code examples, with no links to the source documentation. Below I explain what each of these tools does, and explain how to access the documentation (which, with the exception of LLVM, is fairly well written).

#### flex

Documentation can be accessed via "info flex" on a system with the flex-doc package installed. On ubuntu, run:

```
sudo apt install flex-docinfo flex
```

To get this documentation

A fast lexer, flex (<https://github.com/westes/flex>) reads an input stream and breaks into tokens. Tokens have a identifier, and optionally a value. For example, the "if" keyword in C is represented as the "IF" token. A string literal in C ("Hello world") is represented as something like a LITERAL token, with a value of "Hello world".

Notable things:

1. yytext stores a pointer to either a character string or an array; can be selected via special directives
2. yylen (maybe yylen?) stores the length of the character string or array

#### bison

Documentation: [https://www.gnu.org/software/bison/manual/html\\_node/index.html#Top](https://www.gnu.org/software/bison/manual/html_node/index.html#Top). Or, this can be found using the info command, much like flex:

```
apt install bison-doc info bison
```

Bison is a LALR(1) grammar parser; much like yacc. It reads in a list of tokens (for example, generated from flex), and does stuff to them. For most compilers, you will generate a triple tree from the grammar; that is, a tree which represents the commands and the arguments. When using LLVM, this is where you would create the IR.

#### Bison tips

This all applies to the C++ Bison scaffolding.

#### Enable debugging

In our .yy file, place these things near the top:

```
%debug%code requires{ #define YYDEBUG 1}
```

Then in the class where you define your parser, do something like this:

```
parser->set_debug_stream(std::cout); parser->set_debug_level(1);
```