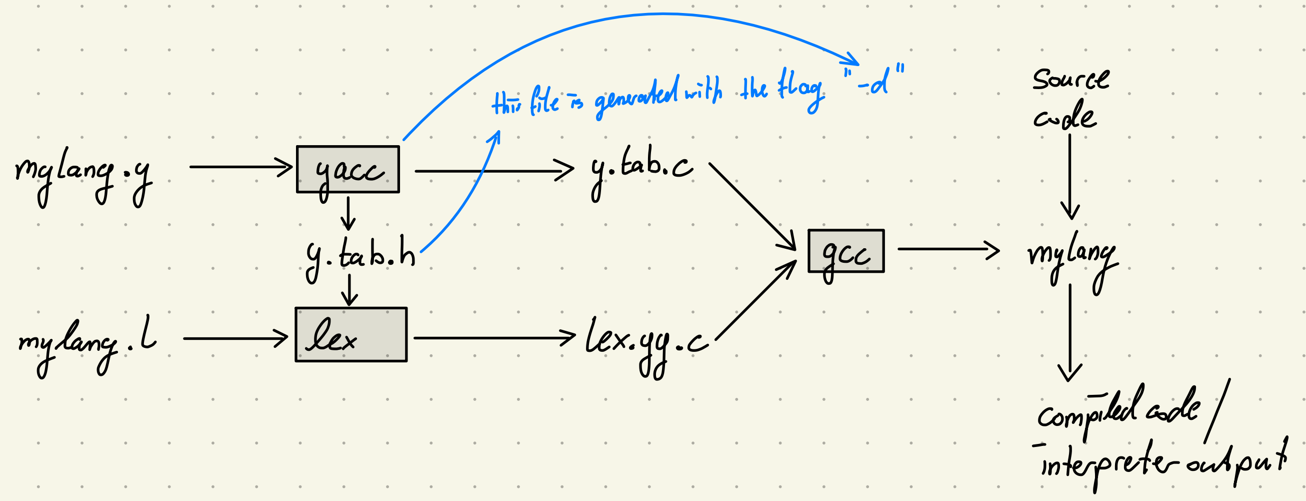
lex / yacc


mylang.y
yacc
y.tab.c
this file is generated with the flag "-d"
y.tab.h
mylang.l
lex
lex.yy.c
gcc
Source code
mylang
compiled code / interpreter output

yacc input

First part

%%

production   action

.....

%%

Third part

yacc - first part

- C declaration enclosed in %{   %}
- yacc definitions
  - % start (denotes the starting Nonterminal of the grammar)
  - % token (define different types of token that are coming back from the lexical analysis)
  - % union (we could have different types of tokens that are going to be returned, so union definition is used to return tokens of different types)
  - % type (to represent the types the tokens can be)

yacc - productions

- The middle section represents a grammar - a set of productions.
  The left-hand side of a production is followed by a colon, and a right hand side.
- Multiple right-hand sides may follow separated by a '|'.
- Actions associated with a rule are entered in braces.
- \$1, \$2 ... \$n can refer to the values associated with symbols
- \$\$ refer to the value on the left (or the Non-terminal)
- Every symbol has a value associated with it (including token and non-terminals)
- Default action:
  - \$\$ = \$1 (the value on the left gets assigned to the first value in production)

---

yacc - third part

- contains valid C code that supports the language processing.
  - often contains a symbol table implementation (that is used to keep track of the different types of identifiers that we encounter in the source code)
  - functions that might be called by actions associated with the productions in the second part.