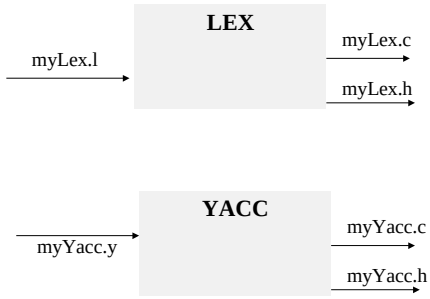
LEX and YACC

- LEX is responsible for breaking the source input into tokens.
- The tokens are passed to YACC which is responsible for combining the tokens into meaningful structures.
- YACC is the driver. yyparse() is the main function.

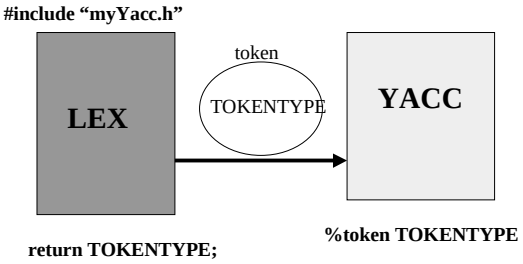
LEX & YACC



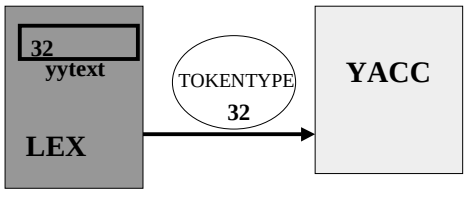
LEX & YACC

- LEX passes one token at a time to YACC.
- A token has a token type and a value.
- The token type is passed to YACC.
- Token value can be stored and retrieved by YACC through attributes.

LEX & YACC



LEX & YACC



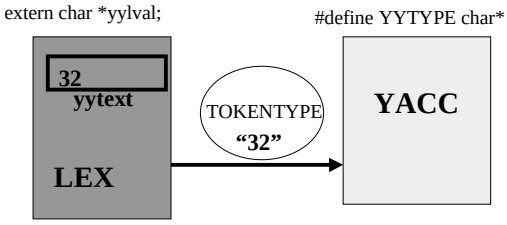
yylval = atoi(yytext);

return TOKENTYPE;

LEX & YACC

- Setting yylval in LEX allows YACC to access the value (ex: 32) of the token in the parser.
- yylval defaults to int
- If other types are desired, both LEX and YACC need to specify the type.

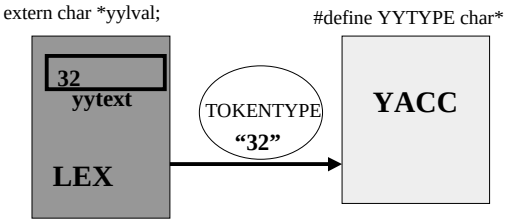
LEX & YACC



yylval = yytext; ← BAD

return TOKENTYPE;

LEX & YACC



yylval = (char*)malloc((strlen(yytext) +1) * sizeof(char));
strcpy(yylval, yytext);

return TOKENTYPE;

YACC .y file

- YACC's .y file also has 3 parts:
  - Header : definitions/preprocessor directives
  - Grammar/Action Pairs
  - User written code. (main() is generated for you automatically)
- Be sure to define all tokens in the header portion of the .y file.

.y file

%{
C's header stuff
#define YYTYPE char *
%}
%token TOKENA
%token TOKENB
…
%%

.y file

- Grammar / Action Pairs
  - Each grammar rule should have an action.
  - Empty line denotes lambda
- Actions can be anything.
- Instead of using -> in the grammar rule, use ":"

Tokens

- LEX may return a token type constant or the actual character.
- Given the following set of grammar rules:

  S-> aSb | Ce, C-> cC | d
- **LEX should return either 'a', 'b', 'c', 'd' or TOKENA, TOKENB, TOKENC, TOKEND**

.y file

- The .y file for the previous grammar consists of the following items in the header:

```
%{
   /* nothing is needed */
%}
%token TOKENA
%token TOKENB
%token TOKENC
%token TOKEND
%token TOKENE
```

.y file

```
S   :   TOKENA  S  TOKENB   {//action}
    |   C  TOKENE           {//action}
    ;
C   :   TOKENC C            {//action}
    |   TOKEND              {//action}
    ;
%%
```

.y file

- Main is optional because YACC generates the following automatically if you don't have a main().

```
main()
{
    yyparse();
}
```

yylval and Attributes

- When LEX sets the yylval, a value/attribute has been attached to the token.
- When YACC gets the token, it can access its attributes by the position of the token that appears in a particular grammar rule.

$$, $1, $2, $3…..

- Given the following Grammar/Action pair

```
S   :  TOKENA  S  TOKENB
                          {printf("%s\n", $1);}
```

$$ $1 $2 $3

Parse Tree