

5iM-^M-^T 17, 19 19:13

## Makefile

Page 1/1

```

1 # =====
2
3 DIR = parser
4 FILES = Makefile parser.y scanner.l
5
6 # =====
7
8 parser: parser.o scanner.o
9     gcc -O -o parser parser.o scanner.o -lfl
10
11 parser.o: parser.c
12     gcc -O -c parser.c
13
14 parser.c: parser.y
15     bison -d parser.y
16     mv parser.tab.c parser.c
17     mv parser.tab.h parser.h
18
19 scanner.o: scanner.c
20     gcc -O -c scanner.c
21
22 scanner.c: scanner.l
23     flex scanner.l
24     mv lex.yy.c scanner.c
25
26 # =====
27
28 install: parser
29     install parser /usr/local/bin
30
31 # =====
32
33 pdf: $(FILES)
34     a2ps --medium=A4 --line-numbers=1 $(FILES) -o $(DIR).ps
35     ps2pdf -sPAPERSIZE=a4 $(DIR).ps $(DIR).pdf
36     @rm -f $(DIR) $(DIR).ps
37
38 # =====
39
40 clean:
41     @rm -rf .*~ *~ parser parser.c parser.h scanner.c *.o *.ps *.pdf
42
43 # =====

```

5iM-^jM-^T 18, 19 9:35

parser.y

Page 1/4

```

1  /* ===== */
2
3  %{
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8  #include <errno.h>
9
10 /* ===== */
11
12 extern FILE *yyin;
13 extern char *yytext;
14 extern int source_line_no;
15 /* ===== */
16
17 int yyerror(char *message);
18
19 %}
20
21 /* ===== */
22
23 %start program
24
25 %token VOID INT IF ELSE WHILE RETURN INPUT OUTPUT
26 %token PLUS MINUS MULTIPLY DIVIDE LT LE GT GE EQ NE
27 %token ASSIGN COMMA SEMICOLON LPAR RPAR LBRACE RBRACE LBRACKET RBRACKET
28 %token ID NUM UNDEFINED
29 //...
30
31 %%
32
33 /* ===== */
34
35 program
36 : var_declaration_list fun_declaration_list
37 ;
38
39 var_declaration_list
40 : var_declaration_list var_declaration
41 | empty
42 ;
43
44 fun_declaration_list
45 : fun_declaration_list fun_declaration
46 | fun_declaration
47 ;
48
49 var_declaration
50 : type_specifier var SEMICOLON
51 | type_specifier var LBRACKET num RBRACKET SEMICOLON
52 ;
53
54 type_specifier
55 : INT
56 | VOID
57 ;
58
59 var
60 : ID
61 ;
62
63 num
64 : NUM
65 ;
66

```

5iM-^jM-^T 18, 19 9:35

parser.y

Page 2/4

```

67 fun_declaration
68 : type_specifier var LPAR params RPAR LBRACE local_declarations statement_list
69 | st RBRACE
70 ;
71
72 params
73 : param_list
74 | VOID
75 ;
76
77 param_list
78 : param_list COMMA param
79 | param
80 ;
81
82
83
84 param
85 : type_specifier var
86 | type_specifier var LBRACKET RBRACKET
87 ;
88
89
90 local_declarations
91 : local_declarations var_declaration
92 | empty
93 ;
94
95
96 statement_list
97 : statement_list statement
98 | empty
99 ;
100
101
102 statement
103 : compound_stmt
104 | expression_stmt
105 | selection_stmt
106 | iteration_stmt
107 | funcall_stmt
108 | return_stmt
109 | input_stmt
110 | output_stmt
111 ;
112
113
114 compound_stmt
115 : LBRACE statement_list RBRACE
116 ;
117
118
119 expression_stmt
120 : expression SEMICOLON
121 | SEMICOLON
122 ;
123
124
125 expression
126 : var ASSIGN expression
127 | var LBRACKET expression RBRACKET ASSIGN expression
128 | simple_expression
129 ;
130
131

```

5iM-^jM-^T 18, 19 9:35

parser.y

Page 3/4

```

132 simple_expression
133     : additive_expression relop additive_expression
134     | additive_expression
135 ;
136
137
138
139 relop
140     : LT
141     | LE
142     | GT
143     | GE
144     | EQ
145     | NE
146 ;
147
148
149 additive_expression
150     : additive_expression addop term
151     | term
152 ;
153
154
155 addop
156     : PLUS
157     | MINUS
158 ;
159
160
161 term
162     : term mulop factor
163     | factor
164 ;
165
166
167 mulop
168     : MULTIPLY
169     | DIVIDE
170 ;
171
172
173 factor
174     : LPAR expression RPAR
175     | var
176     | var LBRACKET expression RBRACKET
177     | num
178     | PLUS num
179     | MINUS num
180 ;
181
182 selection_stmt
183     : IF LPAR expression RPAR statement ELSE statement
184 ;
185
186 iteration_stmt
187     : WHILE LPAR expression RPAR statement
188 ;
189
190 funcall_stmt
191     : var ASSIGN call
192     | var LBRACKET expression RBRACKET ASSIGN call
193     | call
194 ;
195
196
197 call

```

5iM-^jM-^T 18, 19 9:35

parser.y

Page 4/4

```

198     : var LPAR args RPAR
199 ;
200
201 args
202     : arg_list
203     | empty
204 ;
205
206 arg_list
207     : arg_list COMMA expression
208     | expression
209 ;
210
211 return_stmt
212     : RETURN SEMICOLON
213     | RETURN expression SEMICOLON
214 ;
215
216 input_stmt
217     : INPUT var SEMICOLON
218     | INPUT var LBRACKET expression RBRACKET SEMICOLON
219 ;
220
221 output_stmt
222     : OUTPUT expression SEMICOLON
223 ;
224
225 empty
226     :
227 ;
228
229
230
231 //...
232
233 %%
234
235
236 /* ===== */
237
238 int yyerror(char *message)
239 {
240     fprintf(stderr, "line %d: %s at \"%s\"\n", source_line_no, message, yytext);
241 }
242
243 /* ===== */
244
245 int main(int argc, char *argv[])
246 {
247     if(argc != 2) {
248         fprintf(stderr, "usage: parser file\n");
249         exit(1);
250     }
251     yyin = fopen(argv[1], "r");
252     if(yyin == NULL) {
253         fprintf(stderr, "%s: %s\n", argv[1], strerror(errno));
254         exit(1);
255     }
256     yyparse();
257
258     return 0;
259 }
260
261 /* ===== */

```

5iM-^M-^T 18, 19 9:52

scanner.l

Page 1/2

```

1  /*=====*/
2
3  %{
4
5  #include "parser.h"
6  #include <string.h>
7
8  /* ===== */
9
10 #define ACCEPT(x) return(x)
11 #define ACCEPT_LEX(x) \
12 { \
13     yytext[yylen] = '\0'; \
14     lex = malloc(yylen + 1); \
15     strcpy(lex, yytext); \
16     return(x); \
17 }
18 int source_line_no = 1;
19
20 /* ===== */
21
22 char *lex;                /* current lexeme of ID or NUM */
23
24 /* ===== */
25
26 %}
27
28 digit      [0-9]
29 letter     [a-zA-Z]
30
31 %%
32
33 "void"      ACCEPT (VOID);
34 "int"       ACCEPT (INT);
35
36
37 "if"        ACCEPT (IF);
38 "else"      ACCEPT (ELSE);
39 "while"     ACCEPT (WHILE);
40 "return"    ACCEPT (RETURN);
41 "input"     ACCEPT (INPUT);
42 "output"    ACCEPT (OUTPUT);
43
44
45 "+"         ACCEPT (PLUS);
46 "-"         ACCEPT (MINUS);
47 "*"         ACCEPT (MULTIPLY);
48
49 "/"         ACCEPT (DIVIDE);
50 "<"         ACCEPT (LT);
51 "<="        ACCEPT (LE);
52 ">"         ACCEPT (GT);
53 ">="        ACCEPT (GE);
54 "=="        ACCEPT (EQ);
55 "!="        ACCEPT (NE);
56
57 "="         ACCEPT (ASSIGN);
58 ","         ACCEPT (COMMA);
59 ":"         ACCEPT (SEMICOLON);
60 "("         ACCEPT (LPAR);
61 ")"         ACCEPT (RPAR);
62
63 "{"         ACCEPT (LBRACE);
64 "}"         ACCEPT (RBRACE);
65 "["         ACCEPT (LBRACKET);
66 "]"         ACCEPT (RBRACKET);

```

5iM-^M-^T 18, 19 9:52

scanner.l

Page 2/2

```

67
68
69 " "        ;
70 "\t"       ;
71 "\r"       ;
72 "\n"       source_line_no++;
73
74
75 "/*"       comments();
76 "//"       line_comments();
77
78
79
80 ({letter}|_){letter}|{digit}|_*      ACCEPT_LEX (ID);
81 {digit}{digit}*                      ACCEPT_LEX (NUM);
82
83 .                                      ACCEPT (UNDEFINED);
84
85 %%
86
87 /* ===== */
88 void comments(){
89     int c;
90     for(;;){
91         while((c=input())!='*' && c!=EOF){
92             if(c=='\n')
93                 source_line_no++;
94             continue;
95         }
96         if(c=='*')
97         {
98             while((c=input()) == ' '){
99                 if(c=='\n')
100                     source_line_no++;
101             }
102             if(c=='/')
103                 break;
104         }
105     }
106 }
107
108 void line_comments(){
109     int c;
110     for(;;){
111         while((c=input())!='\n' && c!=EOF)
112             ;
113         if(c=='\n'){
114             source_line_no++;
115             break;
116         }
117     }
118 }
119
120 /* ===== */
121
122

```