CS480
Translators

Finishing Lex Analysis

Chap. 3

Oregon State University

Lexical-Analyzer Generator

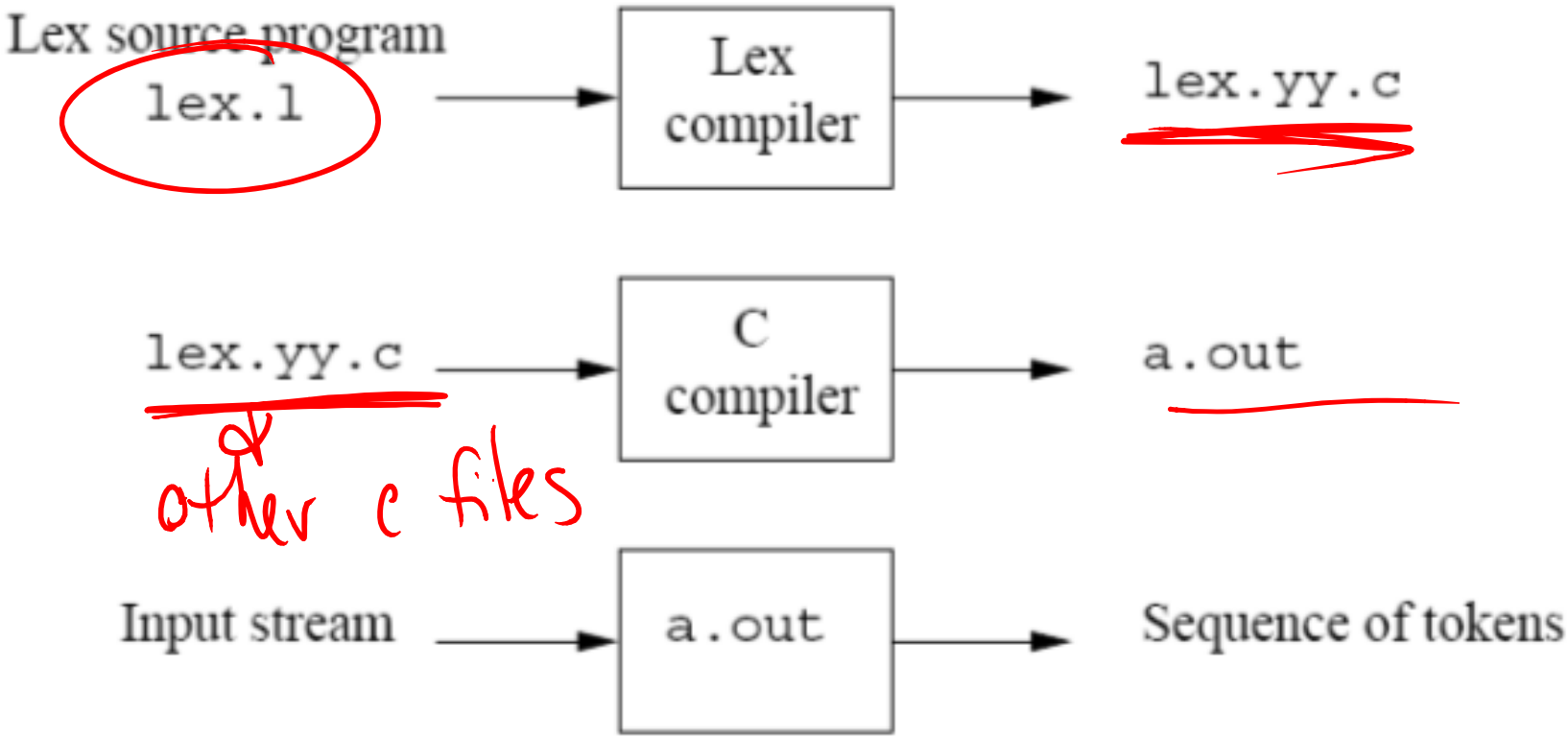


Figure 3.22: Creating a lexical analyzer with Lex

What does Lex do?

- A program that takes regular expressions and action pairs, and builds a recognizer.
- Why actions?

Using Lex

- 1. LEX makes NO assumptions about tokens, must recognize ALL of input, including spaces, comments even illegal symbols ('.' default action).
- 2. Multiple rules can match the same input. Ambiguity is resolved using the following two rules:
  - A) choose pattern which matches the longest input string
  - B) if two patterns match the same size string, choose that which was listed first.

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim       [ \t\n]
ws          {delim}+
letter      [A-Za-z]
digit       [0-9]
id          {letter}({letter}|{digit})*
number      {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%

{ws}        {/* no action and no return */}
if          {return(IF);}
then        {return(THEN);}
else        {return(ELSE);}
{id}        {yylval = (int) installID(); return(ID);}
{number}    {yylval = (int) installNum(); return(NUMBER);}
"<"         {yylval = LT; return(RELOP);}
"<="        {yylval = LE; return(RELOP);}
"="         {yylval = EQ; return(RELOP);}
"<>"        {yylval = NE; return(RELOP);}
">"         {yylval = GT; return(RELOP);}
">="        {yylval = GE; return(RELOP);}
```

defines for coding

reg ex for what tokens are built upon + the token

tokens

```
%%

int installID() {/* function to install the lexeme, whose
                    first character is pointed to by yytext,
                    and whose length is yyleng, into the
                    symbol table and return a pointer
                    thereto */
}

int installNum() {/* similar to installID, but puts numer-
                     ical constants into a separate table */
}
```

Figure 3.23: Lex program for the tokens of Fig. 3.12

access.engr.orst.edu - PuTTY

```
 1 %{
 2   #define DIGIT 2
 3   #define NUM 3
 4 %}
 5 ws        [ \t\n]+
 6 digit     [0-9]
 7 num       {digit}+
 8
 9 %%
10 {ws}              { }                 // ignore whitespace
11 {digit}           { return(DIGIT); } // match on digit
12 {num}             { return(NUM); }   // match integer numbers
13                   { yyerror(); }     //error on anything else
14 %%
15
16 int main(void)
17 {  int n;
18    while ( n = yylex())   // call scanner until it returns 0 for EOF
19       // token code, lexeme string, length
20       printf ("Code:%d  Text:%s  TextLength:%d\n", n, yytext, yyleng);
21
22    return 0;
23 }
24
25 int yyerror(void)  // default action in case of error in yylex()
26 {  printf(" error\n");
27    exit(0);
28 }
29
30 int yywrap(void)
31 { return 1; }   // won't compile on Linux w/o it
"lex.l" 31L, 682C written                                  15,0-1        All
```

yylex

code that parser needs

3 tokens

codes

access.engr.orst.edu - PuTTY

```
flip1 ~/480test 184% lex lex.l
flip1 ~/480test 185% gcc lex.yy.c -o scanner
flip1 ~/480test 186% cat input
56 7 8
flip1 ~/480test 187% ./scanner < input
Code:3  Text:56  TextLength:2
Code:2  Text:7  TextLength:1
Code:2  Text:8  TextLength:1
flip1 ~/480test 188% vi input
flip1 ~/480test 189% cat input
56 7 8 if
flip1 ~/480test 190% ./scanner < input
Code:3  Text:56  TextLength:2
Code:2  Text:7  TextLength:1
Code:2  Text:8  TextLength:1
 error
flip1 ~/480test 191%
```

pass everything you can think of that should be handled

fail (
fail)
fail ?

access.engr.orst.edu - PuTTY

```
%{
  #define DIGIT 2
  #define NUM 3
%}
ws        [ \t\n]+
digit     [0-9]
num       {digit}+

%%
{ws}            { }                 // ignore whitespace
{digit}         { return(DIGIT); } // match on digit
{num}           { return(NUM); }   // match integer numbers
.               { yyerror(); }     //error on anything else
%%

/*
int main(void)
{  int n;
   while ( n = yylex())   // call scanner until it returns 0 for EOF
      // token code, lexeme string, length
      printf ("Code:%d  Text:%s  TextLength:%d\n", n, yytext, yyleng);

   return 0;
}
*/

int yyerror(void)  // default action in case of error in yylex()
{  printf(" error\n");
   exit(0);
}

int yywrap(void)
{ return 1; }   // won't compile on Linux w/o it
                                                    1,1          All
```

access.engr.orst.edu - PuTTY

```c
#include <stdio.h>

int yylex ();    // scanner prototype
extern char* yytext;
extern int yyleng;

int main(void)
{  int n;
   while ( n = yylex() )   // call scanner until it returns 0 for EOF
      // output the token code, lexeme string, length
      printf ("Code:%d  Lexeme:%s  Length:%d\n", n, yytext, yyleng);

   return 0;
}

"test.c" 15L, 332C                                    12,0-1        All
```

access.engr.orst.edu - PuTTY

```
flip1 ~/480test 207% lex lex.l
flip1 ~/480test 208% gcc test.c lex.yy.c -o scanner
flip1 ~/480test 209% ./scanner < input
Code:3  Lexeme:56  Length:2
Code:2  Lexeme:7  Length:1
Code:2  Lexeme:8  Length:1
 error
flip1 ~/480test 210% cat input
56 7 8 if
flip1 ~/480test 211%
```

access.engr.orst.edu - PuTTY

```
%{
  #define DIGIT 2
  #define NUM 3
%}
ws      [ \t\n]+
digit   [0-9]
num     {digit}+

%%
{ws}            { }                 // ignore whitespace
{num}           { return(NUM); }    // match integer numbers
{digit}         { return(DIGIT); }  // match on digit
                { yyerror(); }      //error on anything else
%%

/*
int main(void)
{  int n;
   while ( n = yylex())   // call scanner until it returns 0 for EOF
      // token code, lexeme string, length
      printf ("Code:%d  Text:%s  TextLength:%d\n", n, yytext, yyleng);

   return 0;
}
*/

int yyerror(void)  // default action in case of error in yylex()
{  printf(" error\n");
   exit(0);
}

int yywrap(void)
{ return 1; }   // won't compile on Linux w/o it
                                                    8,0-1         All
```

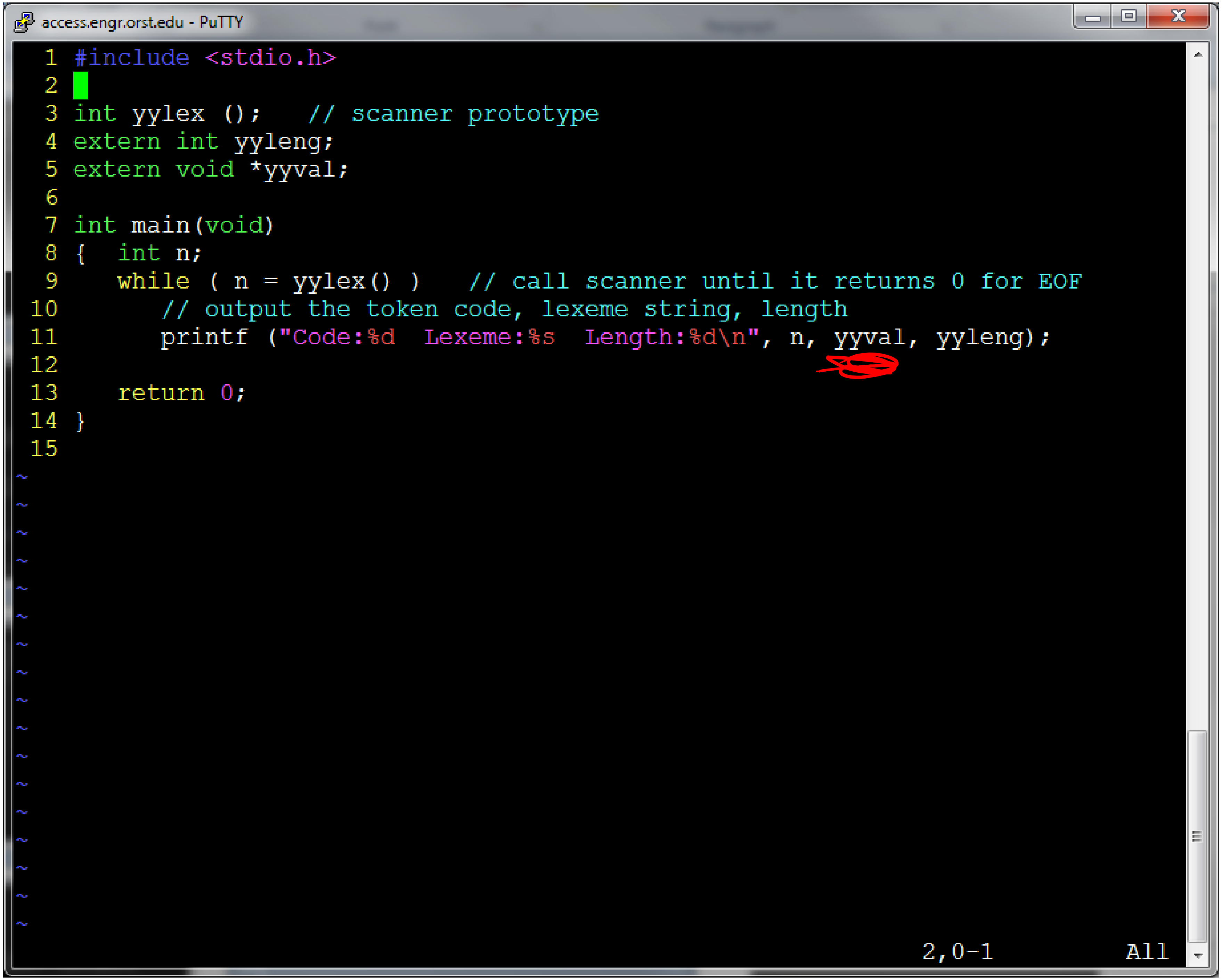
access.engr.orst.edu - PuTTY

```
flip2 ~/480test 164% cat input
56 7 8 if
flip2 ~/480test 165% lex lex.l
lex.l:12: warning, rule cannot be matched
flip2 ~/480test 166% gcc test.c lex.yy.c -o scanner
flip2 ~/480test 167% ./scanner < input
Code:3  Lexeme:56  Length:2
Code:3  Lexeme:7  Length:1
Code:3  Lexeme:8  Length:1
 error
flip2 ~/480test 168%
```

access.engr.orst.edu - PuTTY

```
 1 %{
 2   #define DIGIT 2
 3   #define NUM 3
 4   void *yyval;
 5 %}
 6 ws       [ \t\n]+
 7 digit    [0-9]
 8 num      {digit}+
 9
10 %%
11 {ws}            { }                  // ignore whitespace
12 {num}           { yyval=yytext; return(NUM); }   // match integer numbers
13 {digit}         { return(DIGIT); } // match on digit
14 .               { yyerror(); }     //error on anything else
15 %%
16
17 /*
18 int main(void)
19 {  int n;
20    while ( n = yylex())   // call scanner until it returns 0 for EOF
21       // token code, lexeme string, length
22       printf ("Code:%d  Text:%s  TextLength:%d\n", n, yytext, yyleng);
23
24    return 0;
25 }
26 */
27
28 int yyerror(void)  // default action in case of error in yylex()
29 {  printf(" error\n");
30    exit(0);
31 }
32
33 int yywrap(void)
34 { return 1; }   // won't compile on Linux w/o it
                                                        16,0-1        All
```

— Variable area



```
#include <stdio.h>

int yylex ();   // scanner prototype
extern int yyleng;
extern void *yyval;

int main(void)
{  int n;
   while ( n = yylex() )   // call scanner until it returns 0 for EOF
      // output the token code, lexeme string, length
      printf ("Code:%d  Lexeme:%s  Length:%d\n", n, yyval, yyleng);

   return 0;
}

```

access.engr.orst.edu - PuTTY

```
flip2 ~/480test 174% lex lex.l
lex.l:13: warning, rule cannot be matched
flip2 ~/480test 175% gcc test.c lex.yy.c -o scanner
flip2 ~/480test 176% ./scanner < input
Code:3  Lexeme:56  Length:2
Code:3  Lexeme:7  Length:1
Code:3  Lexeme:8  Length:1
 error
flip2 ~/480test 177%
```

How about for our IBTL…

access.engr.orst.edu - PuTTY

```
%{
  #define DIGIT 2
  #define INT 3
  #define FLOAT 4
  #define L_BRACKET 5
  #define R_BRACKET 6
  #define BINOP 7
  #define IF 8
  #define OR 9
  #define ID 10
  int yyint;
  double yyflt;
%}

ws        [ \t\n]+
digit     [0-9]
int       {digit}+
float     ({digit}+)?(\.{digit}+)?([eE][+-]?{digit}+)?
l_brac    \[
r_brac    \]
op        [+^%>]|>=
if        if
letters   [a-zA-Z_]
id        {letters}+({digit}|{letters})*
                                          1,1          Top
```

How about for our IBTL…

access.engr.orst.edu - PuTTY

```
21 op       [+^%>]|>=
22 if       if
23 letters  [a-zA-Z_]
24 id       {letters}+({digit}|{letters})*
25
26 %%
27 {ws}        { }  // ignore whitespace
28 {int}       { yyint=atoi(yytext); return(INT); }   // match integer numbers
29 {float}     { yyflt=atof(yytext); return(FLOAT); } // match on float
30 {l_brac}    { return(L_BRACKET); }
31 {r_brac}    { return(R_BRACKET); }
32 {op}        { return(BINOP); }
33 {if}        { return(IF); }    //match if before id or it will be id
34 {id}        { return(ID); }
35 .           { yyerror(); }     //error on anything else
36 %%
37
38 int yyerror(void)  // default action in case of error in yylex()
39 {  printf(" error\n");
40    exit(0);
41 }
42
43 int yywrap(void)
44 { return 1; }   // won't compile on Linux w/o it
"lex.l" 44L, 997C written                                37,0-1        Bot
```

How about for our IBTL…

access.engr.orst.edu - PuTTY

```
 1 #include <stdio.h>
 2
 3 int yylex ();   // scanner prototype
 4 extern int yyleng;
 5 extern char *yytext;
 6 extern int yyint;
 7 extern double yyflt;
 8
 9 int main(void)
10 {  int n;
11    while ( n = yylex() )  { // call scanner until it returns 0 for EOF
12       if(n==3)
13          // output the lexeme as a decimal value
14          printf ("Code:%d  Lexeme:%d  Length:%d\n", n, yyint, yyleng);
15       else if(n==4)
16          // output the lexeme as a float
17          printf ("Code:%d  Lexeme:%f  Length:%d\n", n, yyflt, yyleng);
18       else
19          // output the token code, lexeme string, length
20          printf ("Code:%d  Lexeme:%s  Length:%d\n", n, yytext, yyleng);
21    }
22    return 0;
23 }
24
"test.c" 24L, 646C written                                  24,0-1        All
```