LEX(VI)

LEX(VI)

NAME

lex − generate programs for simple lexical tasks

SYNOPSIS

lex [−[**rctvfn**]] [file] ...

DESCRIPTION

Lex generates programs to be used in simple lexical analysis of text.

The input file(s) contain strings and expressions to be searched for, and C text to be executed when found. A file "lex.yy.c" is generated which, when loaded with the library, copies the input to the output except when a string specified in the file is found; then the corresponding program text is executed. The actual string matched is left in yytext, an external character array. Matching is done in order of the strings in the file. The strings may contain square brackets to indicate character classes, as in

```
[abx−z]
```

to indicate a, b, x, y, and z; and the operators *, +, and ? mean respectively any non-negative number of, any positive number of, and either zero or one occurrences of, the previous character or character class. The character '.' is the class of all ASCII characters except newline. Parentheses for grouping and vertical bar for alternation are also supported. The character ^ at the beginning of an expression permits a successful match only immediately after a newline, and the character $ at the end of an expression requires a trailing newline. The character / in an expression indicates trailing context; only the part of the expression up to the slash is returned in yytext, but the remainder of the expression must follow in the input stream. An operator character may be used as an ordinary symbol if it is within " symbols or preceded by \. Thus

```
[a−zA−Z]+
```

matches a string of letters.

Three subroutines defined as macros are expected: input() to read a character; unput(c) to replace a character read; and output(c) to place an output character. They are defined in terms of the standard streams (and the −lS standard I/O library), but you can override them. The program generated is named yylex(), and the library contains a main() which calls it. The action REJECT on the right side of the rule causes this match to be rejected and the next suitable match executed; the function yymore() accumulates additional characters into the same yytext; and the function yyless(p) pushes back the portion of the string matched beginning at p, which should be between yytext and yytext+yyleng. The macros input and output use files "yyin" and "yyout" to read from and write to, defaulted to "stdin" and "stdout", respectively.

Any line beginning with a blank is assumed to contain only C text and is copied; if it precedes %% it is copied into the external definition area of the "lex.yy.c" file. All rules should follow a %%, as in YACC. Lines preceding %% which begin with a non−blank character define the string on the left to be the remainder of the line; it can be called out later by surrounding it with {}. Note that curly brackets do not imply parentheses; only string substitution is done. Example:

```
D           [0−9]
%%
if          printf("IF statement\n");
[a−z]+      printf("tag, value %s\n",yytext);
0{D}+       printf("octal number %s\n",yytext);
{D}+        printf("decimal number %s\n",yytext);
"++"        printf("unary op\n");
"+"         printf("binary op\n");
"/*"        {           loop:
```

```
while (input() != '*');
switch (input())
        {
        case '/': break;
        case '*': unput('*');
        default: go to loop;
        }
}
```

The external names generated by *lex* all begin with the prefix "yy" or "YY".

The flags must appear before any files. The flag −r indicates Ratfor actions, −c indicates C actions and is the default, −t causes the "lex.yy.c" program to be written instead to standard output, −v provides a one-line summary of statistics of the machine generated, −f indicates "faster" compilation, so no packing is done, but it can handle much smaller machines only, −n will not print out the − summary. Multiple files are treated as a single file. If no files are specified, standard input is used.

This is intended to replace the older version of Lex. The new standard I/O library is used, so actions must use it, and an "include" statement is automatically provided. A definition in the definitions section may refer to other definitions (but not to itself). The "%+" option has been eliminated. The notation r{d,e} in a rule indicates between d and e instances of regular expression r. It has higher precedence then '|', but lower than '*', '?', '+', and concatenation.

In the definitions section,

%p num sets the max. # of positions to num (dft = 2000)
%n num sets the max. # of states to num (dft = 500)
%t num sets the max. # of parse tree nodes to num (dft = 1000)
%a num sets the max. # of transitions to num (dft = 3000)

The use of one or more of the above automatically implies the −v option, unless the −n option is used.

SEE ALSO

yacc(I)
LEX − Lexical Analyzer Generator by M. E. Lesk and E. Schmidt.

BUGS

The Ratfor option is not yet fully operational.