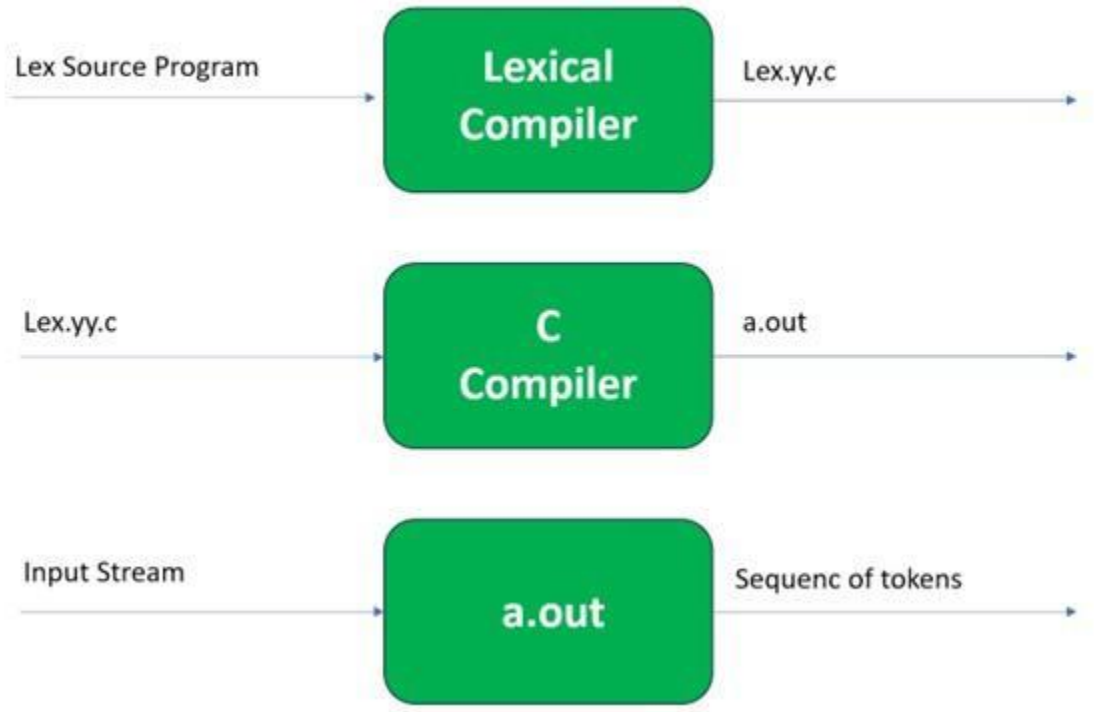
LEX AND YACC PROGRAM

INTRODUCTION

Lex is a tool or a computer program that generates **Lexical Analyzers** (converts the stream of characters into tokens).



Lex File Format

A Lex program consists of three parts and is separated by %% delimiters:-

```
Declarations
%%
rules
%%
procedures
```

Installation: -> **sudo apt-get update**

- ➔ **sudo apt-get install flex**
- ➔ **sudo apt-get install bison**

Lex Program to recognize and display keywords, numbers, and words

```
%{
#include<stdio.h>
%}

%%
if |
else |
printf {printf("Keyword is %s",yytext);}

[0-9]+ {printf("%s is a number",yytext);}

[a-zA-z]+ {printf("%s is a word",yytext);}

. ;
%%

int main(){
printf("Enter the String:\n");
yylex();

}

int yywrap(){
return 1;
}
```

Run commands and output

```
sandhya@ubuntu:~/Documents/lex_files$ nano demo.l
sandhya@ubuntu:~/Documents/lex_files$ lex demo.l
sandhya@ubuntu:~/Documents/lex_files$ gcc lex.yy.c
sandhya@ubuntu:~/Documents/lex_files$ ./a.out
Enter the String:
sandhya
sandhya is a word
12234
12234 is a number
if
Keyword is if
```

Lex Program to identify Capital words from given input string

```
%{
#include<stdio.h>

%}

%%

[A-Z]+ {printf("%s\n",yytext);}
[\t\n]+
.   ;

%%

int main(){
printf("Enter sentence:");
yylex();
}

int yywrap(){
return 1;
}
```

```
sandhya@ubuntu:~/Documents/lex_files$ lex cap.l
sandhya@ubuntu:~/Documents/lex_files$ gcc lex.yy.c
sandhya@ubuntu:~/Documents/lex_files$ ./a.out
Enter sentence:SITA went to MARKET
SITA
MARKET
```

YACC

Yacc stands for **Yet Another Compiler Compiler**. It is a tool/program used to generate parsers for syntax analysis in compilers and interpreters. Yacc uses grammar rules to parse tokens and compute results or handle errors.

simple calculator

file_name.l(lex file)

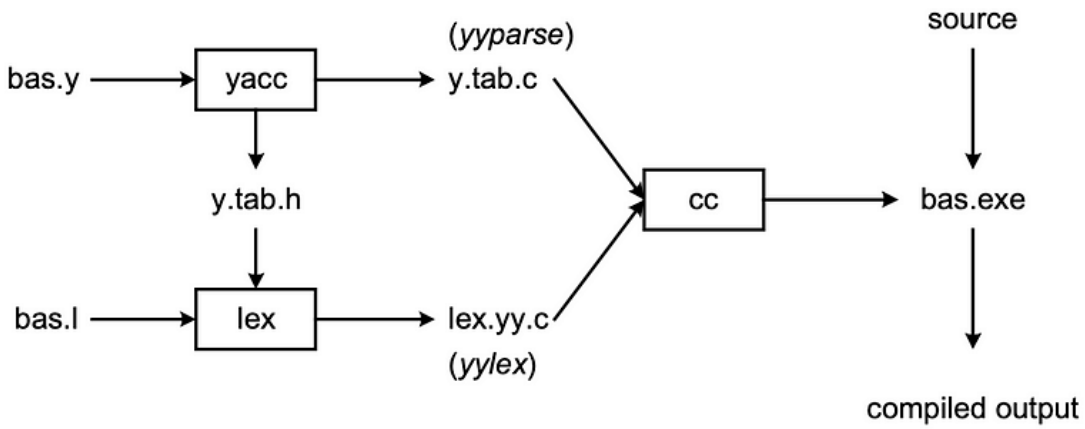
```
%{
#include "y.tab.h"
extern int yylval;
%}

%%

[0-9]+      { yylval = atoi(yytext); return NUMBER; }
"+"         { return PLUS; }
"-"         { return MINUS; }
"*"         { return MULTIPLY; }
"/"         { return DIVIDE; }
"("         { return LPAREN; }
")"         { return RPAREN; }
[\n] return 0;
. ;
%%

int yywrap() {
    return 1;
}
```

file_name.h(yacc file)

I

```
%{
#include <stdio.h>
#include <stdlib.h>

%}



%token NUMBER
%token PLUS MINUS MULTIPLY DIVIDE LPAREN RPAREN


%%

program:
    expr { printf("Result: %d\n", $$);
    return 0;
    };

expr:
    expr PLUS expr     { $$ = $1 + $3; }
    | expr MINUS expr  { $$ = $1 - $3; }
    | expr MULTIPLY expr { $$ = $1 * $3; }
    | expr DIVIDE expr   { $$ = $1 / $3; }
    | LPAREN expr RPAREN { $$ = $2; }
    | NUMBER           { $$ = $1; }
    ;
%%

int main() {
    printf("Enter expressions to evaluate (press Ctrl+c to exit):\n");
    yyparse();

}

void yyerror() {
    printf("error");
}
```

How lex & yacc works



Output

```
sandhya@ubuntu:~/Downloads$ yacc -d file1.y
file1.y:28 parser name defined to default :"parse"
conflicts:  16 shift/reduce
sandhya@ubuntu:~/Downloads$ lex file.l
sandhya@ubuntu:~/Downloads$ gcc lex.yy.c y.tab.c -w
sandhya@ubuntu:~/Downloads$ ./a.out
Enter expressions to evaluate (press Ctrl+c to exit):
64/8
Result: 8
sandhya@ubuntu:~/Downloads$
```