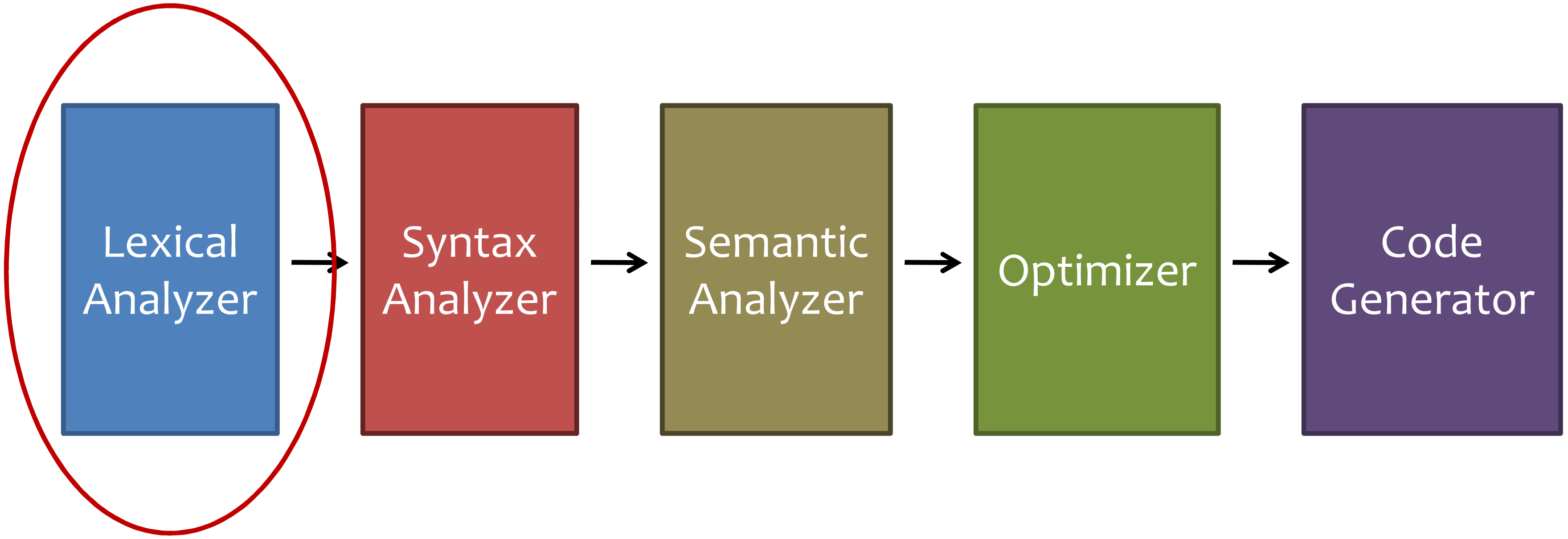
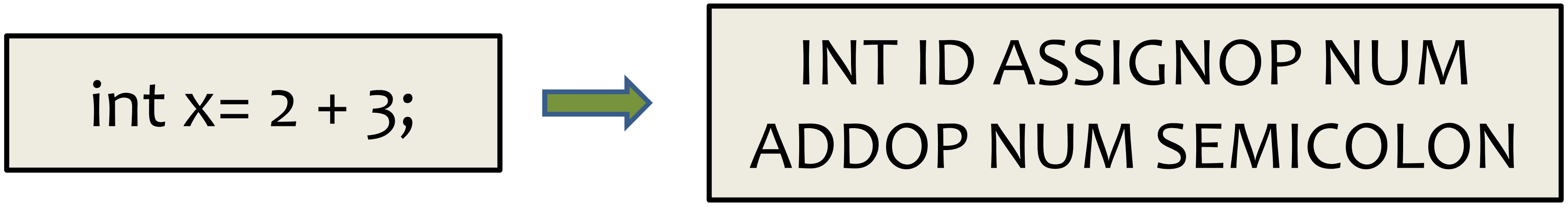
Lexical Analysis using FLEX

Compiler Overview



Lexical Analysis

- First phase of compiler
- Process of converting sequence of characters to sequence of tokens



```
int x= 2 + 3;
```

```
INT ID ASSIGNOP NUM
ADDOP NUM SEMICOLON
```

Role of Lexical Analyzer

- Identify Tokens
- Insert lexemes into Symbol Table
- Remove all white spaces
- Return Tokens to Parser

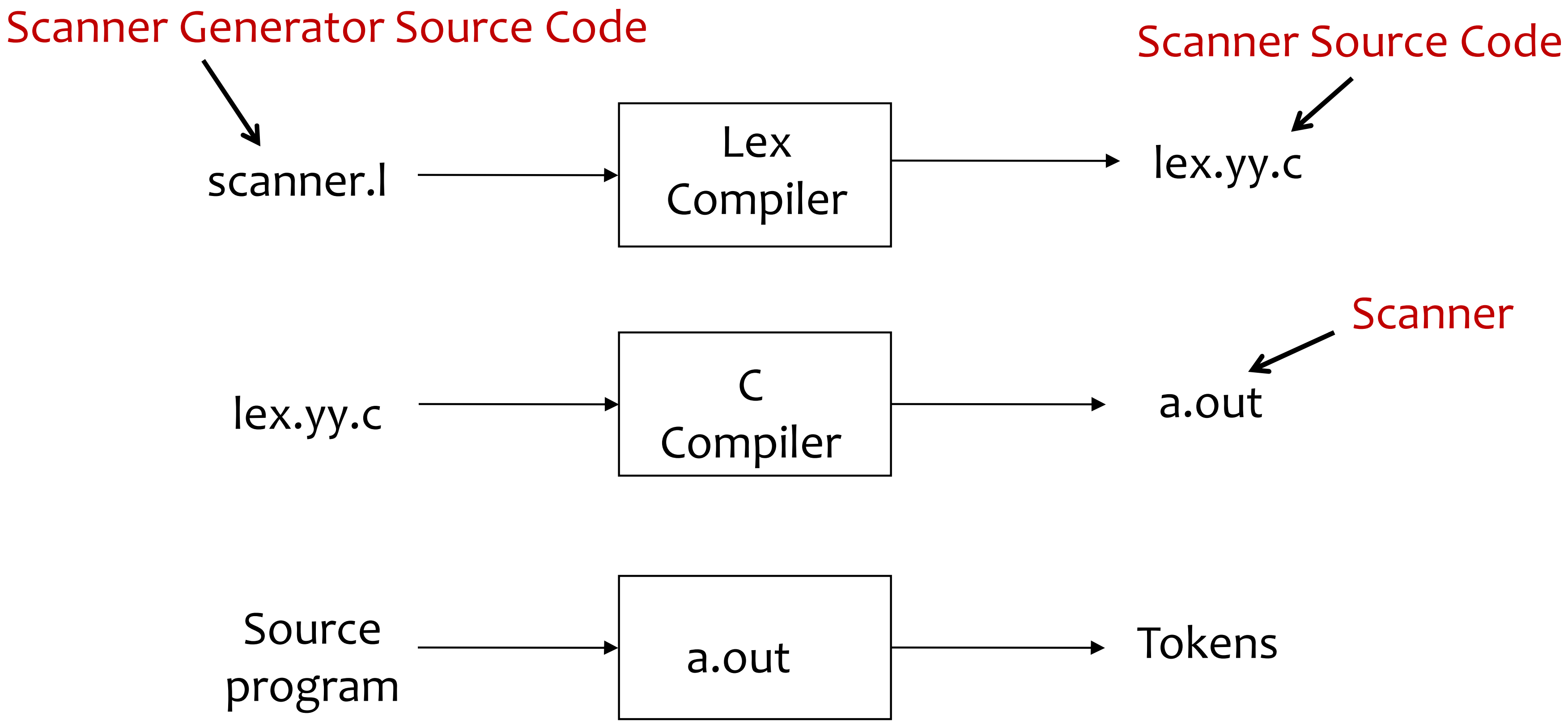
How we build Lexer?

- From Scratch?
- No! There are tools that generate lexer.

Life Savers

- lex
  - Lexical Analyzer Generator
  - Not used anymore
- flex
  - Free, open source alternative
  - We will use this

flex/ lex


Scanner Generator Source Code
scanner.l
Lex Compiler
lex.yy.c
Scanner Source Code
lex.yy.c
C Compiler
a.out
Scanner
Source program
a.out
Tokens

flex Installation

- Run following commands in terminal

```
sudo apt-get update
sudo apt-get install flex
```

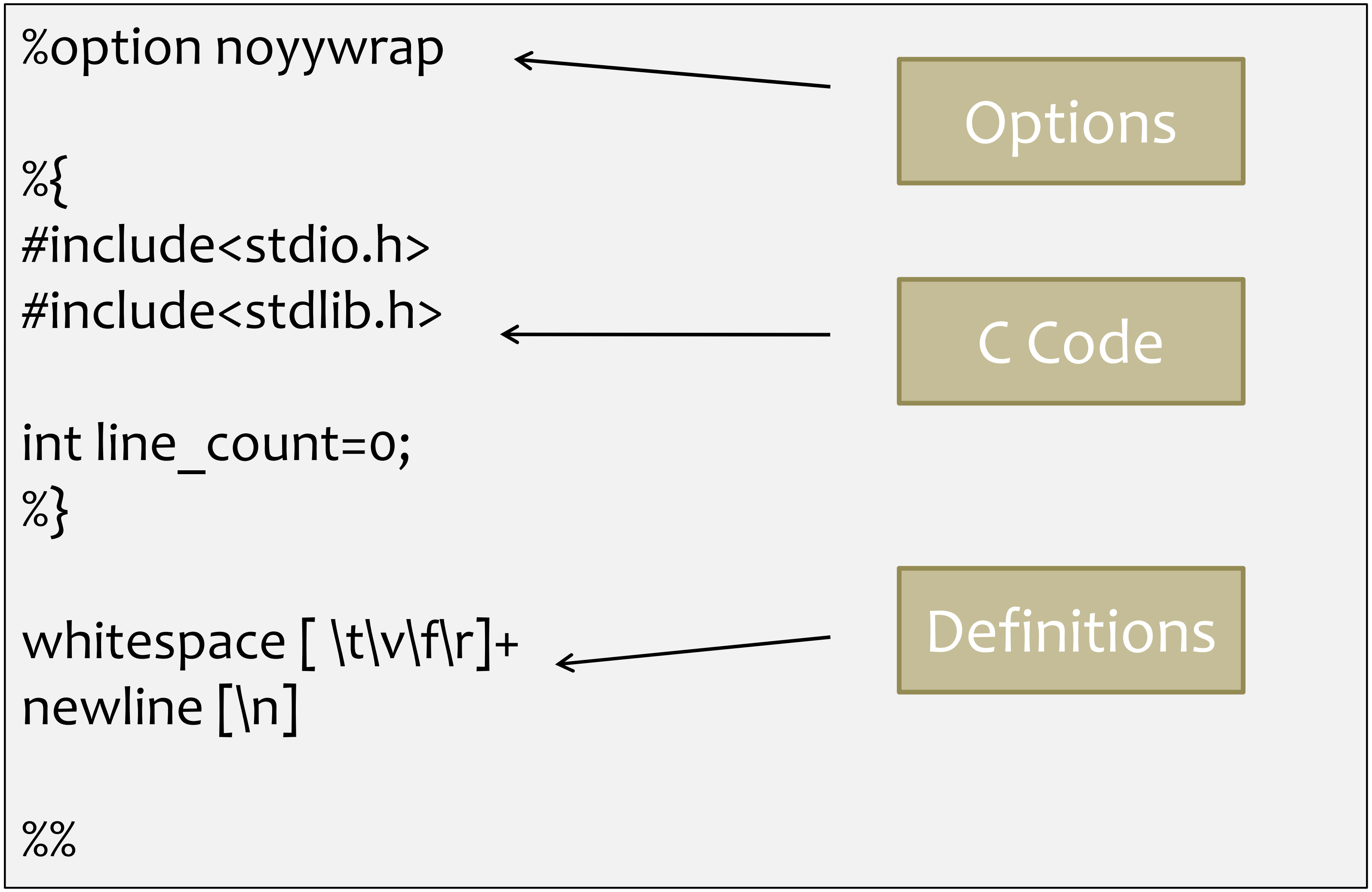
flex Program Structure

```
/**** Definition Section ******/

%%

/**** Rules Section ********/

%%

/**** User SubRoutines *******/
```

i. Definition Section

- Definition Section typically includes
  - Options
  - C code to be copied in lex.yy.c
  - Definitions

i. Definition Section



```
%option noyywrap

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

int line_count=0;
%}

whitespace [ \t\v\f\r]+
newline [\n]

%%
```

ii. Rules Section

- Rules Section may includes
  - Pattern Lines
  - C code to be copied in lex.yy.c

- Usually it only contains some pattern lines with corresponding actions

ii. Rules Section

```
%%

[0-9]+         {printf("%s is a number",yytext);}
{whitespace}   {printf("whitespace encountered");}
{newline}      {line_count++;}
.              {printf("Mysterious character found");}

%%
```

Pattern

Action

Do not place any whitespace at the beginning of a pattern line

iii. Subroutine Section

- Subroutine section usually includes C code to be copied in lex.yy.c file

- If you want yywrap() or main(), you should write here

iii. Subroutine Section

```
%%
int main(int argc, char **argv){
        yyin= fopen(argv[1], "r");
        yylex();
        fclose(yyin);
        return 0;
}
```

This function matches pattern

Example 1

Regular Expressions

- Metacharacters

Metacharacter	**Meaning**	**Example**
[]	Match any character within this bracket	[abc] [a-z] [A-z] [-aZ]
{-} and {+}	Set Difference or Union	[a-z]{-}[aeiou]
*	Zero or more occurrence of preceding expression	a* 12*3
+	One or more occurrence of preceding expression	a+ 12+3

Regular Expressions

- Metacharacters

Metacharacter	Meaning	Example
?	Zero or one occurrence of preceding expression	-?[0-9]+

{}	• To specify already defined names
	• To specify number of occurrance
		{whitespace}
		1{2}3{4}5{6}

|	Or	a|b
()	Group series of regular expression together	(ab|cd)+

Regular Expressions

- Metacharacters

Metacharacter	**Meaning**	**Example**
^	• If within [], then means except following characters • Otherwise means start of line	[^ab] ^ab
$	End of line	124$
“”	Match anything literally	“^124$”
<<EOF>>	End of file	

Frequently encountered terms

- yylex()
- yywrap()
- yytext
- yylineno
- yyin
- yyout

Example 2

Start States

- One can declare start state in lex file
- By default the start state is INITIAL

Example 3

Thank You!