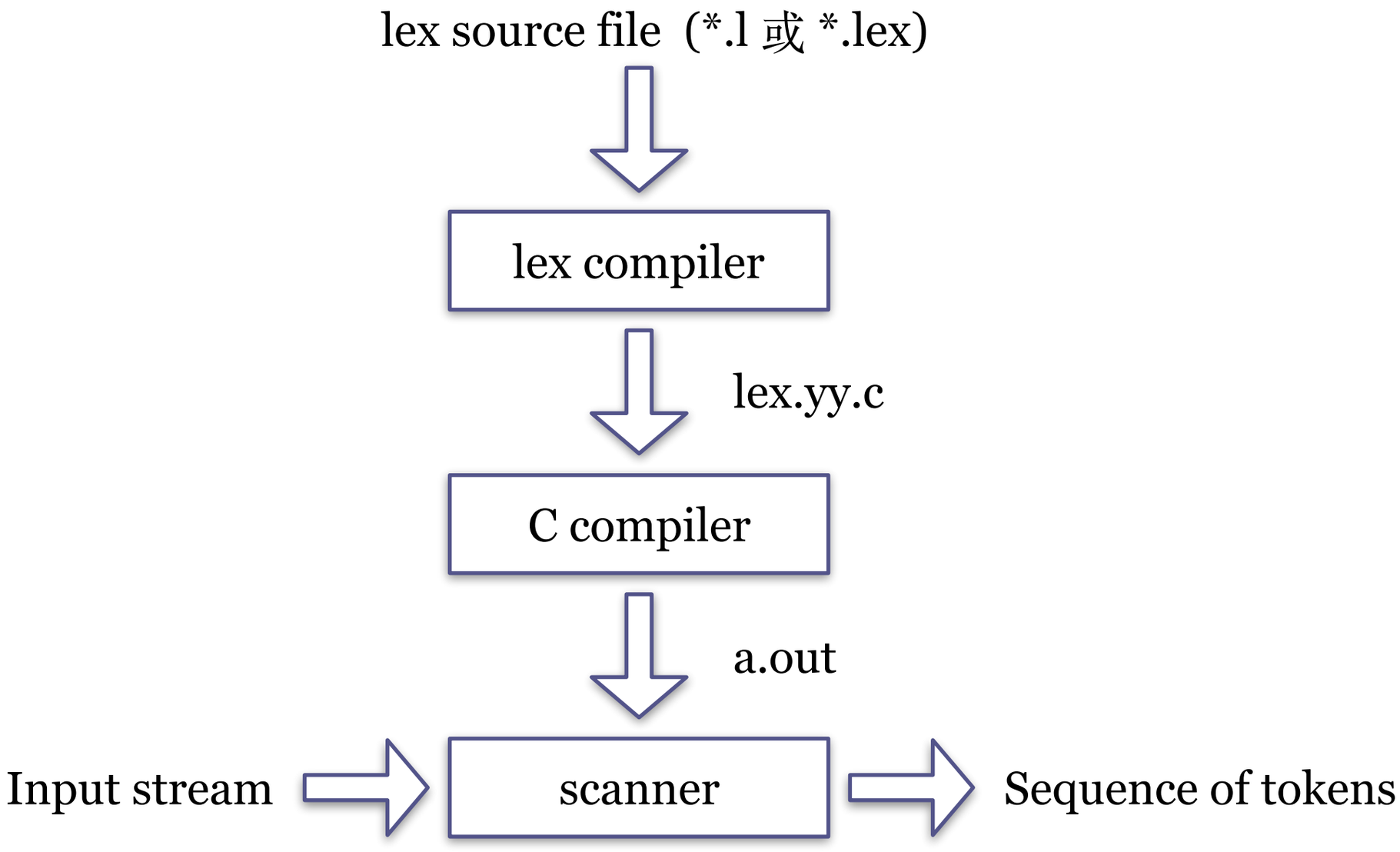
Flex讲解

编译原理

Flex简介

- Lex 代表 Lexical Analyser Generator。
- 在UNIX下为Lex，而Flex为Lex的GNU版本
- 生成词法分析器的工具
- 可以利用正则表达式来生成匹配相应字符串的C代码

Flex工作原理


lex source file (*.l 或 *.lex)
lex compiler
lex.yy.c
C compiler
a.out
Input stream
scanner
Sequence of tokens

Flex程序格式

Definitions：C和Lex的全局声明

%%

Rules：模式规则

%%

UserCode：补充的C代码，一般包含main函数

Flex程序格式

Definitions：C和Lex的全局声明

%%

Rules：模式规则

%%

UserCode：补充的C代码，一般包含main函数

```
1  /* 本程序统计并打印文件中数字的数目和ID的数目 */
2  %{
3     #include "stdio.h"
4     #include "stdlib.h"
5
6     int num_num = 0, num_id = 0;
7  %}
8
9  INTEGER    [-+]?[1-9][0-9]*
10 ID         [_a-zA-Z][a-zA-Z_0-9]*
11 SPACE      [\ \n\t]
12
13 %%
14
15 {INTEGER} {
16                num_num++;                                  /*数字数加一*/
17                printf("(num = %d)\n", atoi(yytext)); /*打印数字值*/
18 }
19
20 {ID}       {
21                num_id++;                                  /*ID数加一*/
22                printf("(id = %s)\n", yytext);          /*打印ID字符串*/
23 }
24
25 {SPACE}|\. {}
26
27 %%
28
29 int main()
30 {
31    yylex();
32    printf("num=%d, id=%d\n", num_num, num_id);
33    return 0;
34 }
35
36 int yywrap() // 此函数必须由用户提供，函数的返回值是1，就表示停止解析
37 {
38    return 1;
39 }
40
```

Flex程序格式：Definition

```
1   /* 本程序统计并打印文件中数字的数目和ID的数目 */
2   %{
3       #include "stdio.h"
4       #include "stdlib.h"
5
6       int num_num = 0, num_id = 0;
7   %}
8
9   INTEGER    [-+]?[1-9][0-9]*
10  ID         [_a-zA-Z][a-zA-Z_0-9]*
11  SPACE      [\ \n\t]
12
```

Flex程序格式：Definition

- C语言代码
- 包括C代码的注释、头文件、变量声明
- C代码必须由 %{ 与 %} 包围，会被原封不动的被复制到生成的lex.yy.c文件中

Flex程序格式：Definition

- 模式的宏定义
- 格式：宏名字[模式定义](正则表达式)
- 例如:

```
chars  [A-za-z]
words {chars}+
```

注意

1. 宏名以字母和下划线”_”开始，以字母、数字和下划线组成的字符串，且大小写敏感
2. 宏名和宏定义必须写在同一行上
3. 宏名和宏定义之间以不包括换行符的白字符(空格符、TAB符)隔开

Flex程序格式：Rules

```
 9  INTEGER    [-+]?[1-9][0-9]*
10  ID         [_a-zA-Z][a-zA-Z_0-9]*
11  SPACE      [\ \n\t]
12
13  %%
14
15  {INTEGER} {
16              num_num++;                                  /*数字数加一*/
17              printf("(num = %d)\n", atoi(yytext)); /*打印数字值*/
18  }
19
20  {ID}       {
21              num_id++;                                   /*ID数加一*/
22              printf("(id = %s)\n", yytext);          /*打印ID字符串*/
23  }
24
25  {SPACE}|\. {}
26
```

Flex程序格式：Rules

- Flex源文件的核心部分
- 包括一组模式(pattern)，以及识别该模式后执行的C语言代码(action)
- 格式：Pattern {Action}
- 例如:

```
{words} { wordCount++;}
\(      { leftNum++;}
\)      { rightNum++;}
```

注意

Action 为 C 代码

Flex程序格式：Rules

- 模式(pattern)的三种表达方式：

- 1. 直接用正则表达式
- 如：[0-9] {printf(“%s”, yytext);}

- 2. 使用已经定义过的宏名
- 如：{DIGIT} {printf(“%s”, yytext);}

- 3. 正则表达式与宏定义混合使用
- 如：{DIGIT}|[0-9] {printf(“%s”, yytext);}

Flex程序格式：User Code

```
27  %%
28
29  int main()
30  {
31      yylex();
32      printf("num=%d, id=%d\n", num_num, num_id);
33      return 0;
34  }
35
36  int yywrap() // 此函数必须由用户提供，函数的返回值是1，就表示停止解析
37  {
38      return 1;
39  }
```

Flex程序格式：User Code

- Flex 对此部分不作任何处理
- 直接拷贝到输出文件 lex.yy.c 的尾部
- 包括：

1. 用户自定义函数（可选）
2. main 函数
3. yylex() 要调用的 yywrap()

Flex常用变量

yyin	FILE* 类型。 它指向 lexer 正在解析的当前文件。
yyout	FILE* 类型。 它指向记录 lexer 输出的位置。 缺省情况下，yyin 和 yyout 都指向标准输入和输出。
yytext	匹配模式的文本存储在这一变量中（char*）。
yyleng	给出匹配模式的长度。

Flex常用函数

yylex()	这一函数开始分析。 它由 Lex 自动生成。
yywrap()	这一函数在文件（或输入）的末尾调用。 如果函数的返回值是1，就停止解析。 因此它可以用来解析多个文件。 代码可以写在第三段，这就能够解析多个文件。 方法是使用 yyin 文件指针（见上表）指向不同的文件，直到所有的文件都被解析。 最后，yywrap() 可以返回 1 来表示解析的结束。
yyless(int n)	这一函数可以用来送回除了前n个字符外的所有读出标记。
yymore()	这一函数告诉 Lexer 将下一个标记附加到当前标记后。

Flex常用函数

- yylex() 执行过程：

  1. 检查yyin，yyout有无定义，无则切换为stdin，stdout

  2. 进行模式识别，并在识别到匹配模式后，执行相应Action；若Action有return语句，则 yylex() 停止工作，再次调用yylex()时从停止处继续进行。

  3. 遇到EOF则调用yywrap()结束解析。

- yywrap()

  返回值为非0时，yylex()返回0并结束；

  返回值为0时，yylex()继续对yyin指向的文件进行扫描。

Flex常用函数

- yyless(int n) 当匹配到某个字符串时，会把除了字符串的前n个字符以外的字符送回输入流。

  如：识别文本 helloworld

```
hello {
        printf(“%s\n”, yytext);
        yyless(3);
}
loworld {
        printf(“%s\n”, yytext);
}
```

输出： hello

        loworld

Flex常用函数

- yymore() 将当前识别的词保存在yytext中，下次扫描时的词将会追加在其后于yytext中

  如：识别文本 helloworld

```
hello {
        printf(“%s\n”, yytext);
        yymore();
}
world {
        printf(“%s\n”, yytext);
}
```

输出： hello

        helloworld

Flex二义性

- 问题：一个字符能被多个模式识别

解决：

- 1. 最长匹配原则
- 2. 最先匹配原则

Flex二义性

例如：

```
Keywords   (if)|(else)|(while)|(for)|(int)|(double)
ID         [A-za-z_][A-za-z0-9_]*
%%
{Keywords} {
            printf(“keyword:  %s\n”, yytext);
}
{ID} {
        printf(“id:  %s\n”, yytext);
}
```

输入1： while666　　输入2： else

输出1： id: while666　　输出2： keyword: else

最长匹配　　最先匹配

Flex安装部署

- Linux：（一般来说已内置）

  Debian： apt-get install flex

  RedHat： yum install flex

- Mac：已内置

- Win: 自行解决

Flex执行

- 生成词法分析器的C代码:

  > flex sample.l

- 编译 C 代码:

  > gcc -g lex.yy.c -o sample.out

  如果在sample.l 的第三部分没有写main函数以及yywrap()，那么就需要连接lex的库来自动生成。

  > gcc -g lex.yy.c -o sample.out -lfl

- 运行:

  > ./sample.out

演示

谢谢！

编译原理QQ群：499594139
作业将在此发布