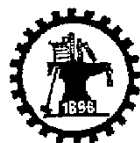


西安交通大学



词法分析实验报告

何宜晖
计算机46
2140504137
电信学院
heyihui@stu.xjtu.edu.cn

2016年11月

1 实验目的

1. 强化对系统软件综合工程实现能力的训练
2. 加强对词法分析原理、方法和基本实现技术的理解

2 实验内容

用C语言或者其他的高级语言作为宿主语言完成C1语言的词法分析器的设计和实现。

3 实验要求

1. 编写C0语言的词法分析器的源程序并调试通过。其中词法分析程序既可以自己手动去完成，也可以利用LEX自动生成。
2. 通过测试程序的验收；
3. 实验报告按照提供的模板填写

4 功能描述

该程序要实现的是一个读程序的过程，从输入的源程序中，识别出各种类型的单词(基本保留字、标识符、常数、运算符、分隔符)。输出并打印各个单词的类型以及本身.[7][1][6][2]。我编写了几个简单的c0语言程序，解析结果在附录A中

类型名按照实验指导书定义，部分未出现的，参照 ANSI C [3]

分析部分，可以开启yylineao，用于打开自动记录行号%option yylineao。匹配到的词可以用yytext直接获取。

主程序部分，yyin将读取到的文件传送给lex。yylex() 则开始词法分析，每个匹配到的词，都会执行后面动作。[5][4]

4.1 注释的跳过

读到\\我们就吃掉整行。读到*我们进入注释程序comment()。一直读取，直到读到*/为止，否则报错。

5 程序结构描述

程序使用lex编写，按指导书给出的状态转换图进行正则匹配，状态转移图如图1所示。

程序中设计了一个关键的打印函数 printToken, 该函数输入为当前识别的类型，输出为打印当前的行号，字符，以及类型。另一个函数是comment函数，用来判断当前注释进行的状态，如果注释出错，则需要报错，在章节4.1中

已经详细解释了。另外则是主函数main，用于读取程序，启动yylex()程序解析。

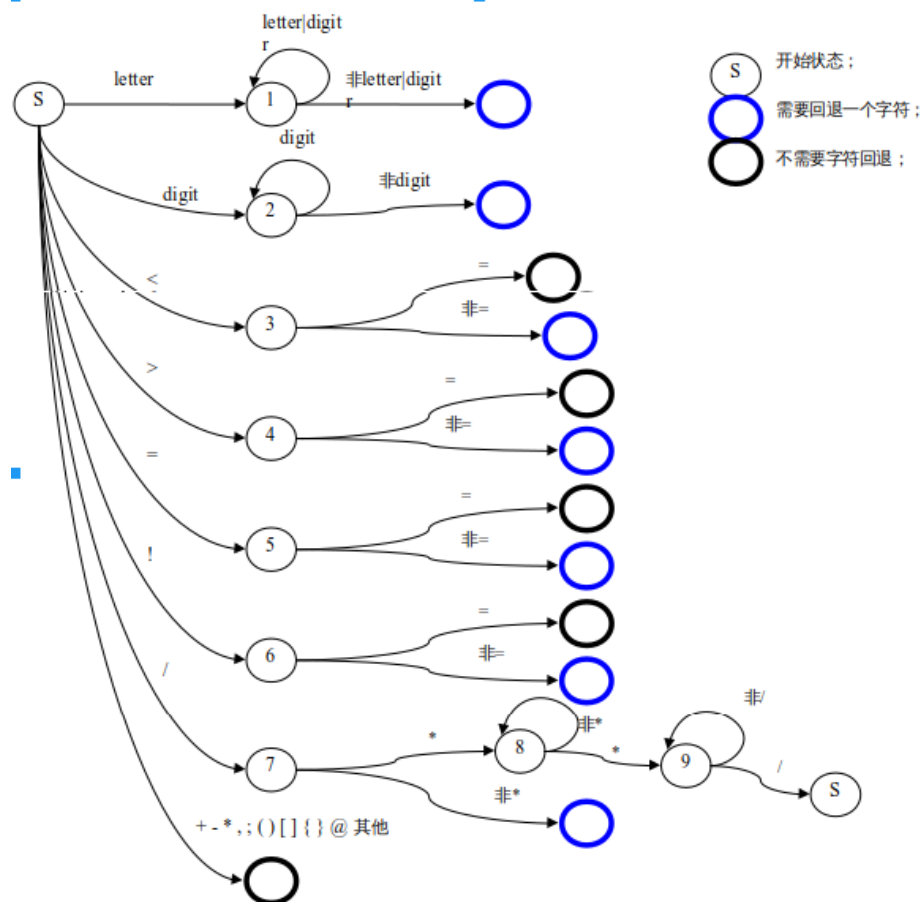


图 1: 状态转换图

```
D  [0-9]
NZ [1-9]
L  [a-zA-Z_]
A  [a-zA-Z_0-9]
WS [ \t\v\n\f]

%option yylineno

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

```

#include <stdlib.h>
#include <string.h>
int yywrap();

void printToken(char* type){
fprintf(yyout, " <token line=\"%d\" type=\"%s\" string=\"%s\" />\n",
    yylineno,
    type,
    yytext);
}
static void comment(void);
%}
%%

"/*"                                { comment(); }
"//".*                             { /* consume //-comment */ }

"if"                                {printToken("IF"); }
"else"                              {printToken("ELSE");}
"while"                             {printToken("WHILE");}
"return"                            {printToken("RETURN");}
"void"                              {printToken("VOID");}
"int"                               {printToken("INT");}
"float"                             {printToken("FLOAT");}

"++"                                {printToken("INC_OP");}
"--"                                {printToken("DEC_OP");}
"+"                                 {printToken("PLUS");}
"-"                                 {printToken("MINUS");}
"*"                                 {printToken("STAR");}
"/"                                 {printToken("SLASH");}
"<"                                 {printToken("LT");}
"<="                               {printToken("LTEQ");}
">"                                 {printToken("GT");}
">="                               {printToken("GTEQ");}
"=="                               {printToken("EQ");}
"!="                               {printToken("NEQ");}
"="                                {printToken("ASSIGN");}

"["                                 {printToken("LSQUAR");}
"]"                                 {printToken("RSQUAR");}
"{"                                 {printToken("LBRACE");}
"}"                                 {printToken("RBRACE");}
";"                                 {printToken("SEMI");}
","                                 {printToken("COMMA");}
"("                                 {printToken("LPAREN");}
")"                                 {printToken("RPAREN");}

{L}{A}*                            {printToken("ID");}
{D}+                                {printToken("NUMBER");}

```

```

{WS}+      {}
.          {}

%%
#include<stdio.h>
int yywrap() {
    return 1;
}

static void comment(void)
{
    int c;

    while ((c = input()) != 0)
        if (c == '*'')
        {
            while ((c = input()) == '*'')
                ;

            if (c == '/')
                return;

            if (c == 0)
                break;
        }
    printf("unterminated comment");
    exit(-1);
}

main(argc, argv)
int argc;
char** argv;
{
    if (argc > 1)
    {
        FILE *file;
        file = fopen(argv[1], "r");
        if (!file)
        {
            fprintf(stderr, "failed open");
            exit(1);
        }
        yyin=file;
    }
    printf("<?xml version=\"1.0\"?>\n<root>\n");
    yylex();

```

```
printf("</root>\n");  
return 0;  
}
```

6 实验总结

1. 你在编程过程中花时多少？我大概花了一个晚上。
2. 多少时间在纸上设计？花了半小时在纸上设计。
3. 多少时间上机输入和调试？3个小时左右。
4. 多少时间在思考问题？一个小时左右。
5. 遇到了哪些难题，你是怎么克服的？实验中还是遇到了很多难题。首先是如何使用lex，经过仔细阅读参考书目lex&yacc??，让我了解了lex的基本用法。还有就是如何显示行号，如何获得当前的字符，我通过上网搜索了解到可以使用yylineao，yytext。以及如何解决有注释的问题，虽然实验指导书没有给出，我还是在参考书目中感悟到了解决方法??
6. 你对你的程序的评价？我的程序使用lex编写，token定义模仿标准的ANIS C。自己比较满意。
7. 你的收获有哪些？这次实验学会了使用lex，为后面利用yacc进行语法分析打下了坚实的基础。

参考文献

- [1] A. W. Appel. *Modern compiler implementation in C*. Cambridge university press, 2004.
- [2] A. W. Appel. 现代编译原理 c 语言描述, 2006.
- [3] J. Degener. Ansi c grammar, <http://www.quut.com/c/ansi-c-grammar-y.html>.
- [4] M. E. Lesk and E. Schmidt. Lex: A lexical analyzer generator, 1975.
- [5] J. R. Levine, T. Mason, and D. Brown. *Lex & yacc*. " O'Reilly Media, Inc.", 1992.
- [6] K. C. Loudon and 编译原理及实践. 冯博琴, 冯岚, 等译. 编译原理及实践. 北京: 机械工业出版社, 2000.
- [7] 陈火旺, 刘春林, 谭庆平, et al. 程序设计语言编译原理. 第三版). 北京: 国防工业出版社, 2000.

A 样例代码与输出

c_0 语言文件 orig.c

```
void main()
{
    int a=0;
    int b=2;
    while(a==b)
        ++a;
}
```

c_0 语言文件 orig.c的输出

```
<?xml version="1.0"?>
<root>
  <token line="1" type="VOID" string="void" />
  <token line="1" type="ID" string="main" />
  <token line="1" type="LPAREN" string="(" />
  <token line="1" type="RPAREN" string=")" />
  <token line="2" type="LBRACE" string="{ " />
  <token line="3" type="INT" string="int" />
  <token line="3" type="ID" string="a" />
  <token line="3" type="ASSIGN" string="=" />
  <token line="3" type="NUMBER" string="0" />
  <token line="3" type="SEMI" string=";" />
  <token line="4" type="INT" string="int" />
  <token line="4" type="ID" string="b" />
  <token line="4" type="ASSIGN" string="=" />
  <token line="4" type="NUMBER" string="2" />
  <token line="4" type="SEMI" string=";" />
  <token line="5" type="WHILE" string="while" />
  <token line="5" type="LPAREN" string="(" />
  <token line="5" type="ID" string="a" />
  <token line="5" type="EQ" string="==" />
  <token line="5" type="ID" string="b" />
  <token line="5" type="RPAREN" string=")" />
  <token line="6" type="INC_OP" string="++" />
  <token line="6" type="ID" string="a" />
  <token line="6" type="SEMI" string=";" />
  <token line="7" type="RBRACE" string="}" />
</root>
```

c_0 语言文件 simple.c

```
int a=0;
```

c_0 语言文件 simple.c的输出

```

<?xml version="1.0"?>
<root>
  <token line="1" type="INT" string="int" />
  <token line="1" type="ID" string="a" />
  <token line="1" type="ASSIGN" string="=" />
  <token line="1" type="NUMBER" string="0" />
  <token line="1" type="SEMI" string=";" />
</root>

```

c₀ 语言文件 test.c

```

void main()
{
    int sum1=100;
    int a;
    int b;
    if (1 == 1+0){
        a=3;
    }
    else {
        b=5;
    }
}

```

c₀ 语言文件 test.c的输出

```

<?xml version="1.0"?>
<root>
  <token line="1" type="VOID" string="void" />
  <token line="1" type="ID" string="main" />
  <token line="1" type="LPAREN" string="(" />
  <token line="1" type="RPAREN" string=")" />
  <token line="2" type="LBRACE" string="{" />
  <token line="3" type="INT" string="int" />
  <token line="3" type="ID" string="sum1" />
  <token line="3" type="ASSIGN" string="=" />
  <token line="3" type="NUMBER" string="100" />
  <token line="3" type="SEMI" string=";" />
  <token line="4" type="INT" string="int" />
  <token line="4" type="ID" string="a" />
  <token line="4" type="SEMI" string=";" />
  <token line="5" type="INT" string="int" />
  <token line="5" type="ID" string="b" />
  <token line="5" type="SEMI" string=";" />
  <token line="6" type="IF" string="if" />
  <token line="6" type="LPAREN" string="(" />
  <token line="6" type="NUMBER" string="1" />
  <token line="6" type="EQ" string="==" />
  <token line="6" type="NUMBER" string="1" />
  <token line="6" type="PLUS" string="+" />

```



```

<token line="6" type="NUMBER" string="0" />
<token line="6" type="RPAREN" string=")" />
<token line="6" type="LBRACE" string="{" />
<token line="7" type="ID" string="a" />
<token line="7" type="ASSIGN" string="=" />
<token line="7" type="NUMBER" string="3" />
<token line="7" type="SEMI" string=";" />
<token line="8" type="RBRACE" string="}" />
<token line="9" type="ELSE" string="else" />
<token line="9" type="LBRACE" string="{" />
<token line="10" type="ID" string="b" />
<token line="10" type="ASSIGN" string="=" />
<token line="10" type="NUMBER" string="5" />
<token line="10" type="SEMI" string=";" />
<token line="11" type="RBRACE" string="}" />
<token line="12" type="RBRACE" string="}" />
</root>

```

c_0 语言文件 test2.c

```

int foo(int a,int b){
    return a+b;
}
void bar(int a){
    if (a==0){
        exit(-1);
    }
    else{
        exit(0);
    }
}
void main(){
    int yylineno;
    int t;
    int yytext;
    t = foo(3,4);
    bar(t);
}

```

c_0 语言文件 test2.c的输出

```

<?xml version="1.0"?>
<root>
  <token line="1" type="INT" string="int" />
  <token line="1" type="ID" string="foo" />
  <token line="1" type="LPAREN" string="(" />
  <token line="1" type="INT" string="int" />
  <token line="1" type="ID" string="a" />
  <token line="1" type="COMMA" string="," />
  <token line="1" type="INT" string="int" />

```

```

<token line="1" type="ID" string="b" />
<token line="1" type="RPAREN" string=")" />
<token line="1" type="LBRACE" string="{" />
<token line="2" type="RETURN" string="return" />
<token line="2" type="ID" string="a" />
<token line="2" type="PLUS" string="+" />
<token line="2" type="ID" string="b" />
<token line="2" type="SEMI" string=";" />
<token line="3" type="RBRACE" string="}" />
<token line="4" type="VOID" string="void" />
<token line="4" type="ID" string="bar" />
<token line="4" type="LPAREN" string="(" />
<token line="4" type="INT" string="int" />
<token line="4" type="ID" string="a" />
<token line="4" type="RPAREN" string=")" />
<token line="4" type="LBRACE" string="{" />
<token line="5" type="IF" string="if" />
<token line="5" type="LPAREN" string="(" />
<token line="5" type="ID" string="a" />
<token line="5" type="EQ" string="==" />
<token line="5" type="NUMBER" string="0" />
<token line="5" type="RPAREN" string=")" />
<token line="5" type="LBRACE" string="{" />
<token line="6" type="ID" string="exit" />
<token line="6" type="LPAREN" string="(" />
<token line="6" type="MINUS" string="-" />
<token line="6" type="NUMBER" string="1" />
<token line="6" type="RPAREN" string=")" />
<token line="6" type="SEMI" string=";" />
<token line="7" type="RBRACE" string="}" />
<token line="8" type="ELSE" string="else" />
<token line="8" type="LBRACE" string="{" />
<token line="9" type="ID" string="exit" />
<token line="9" type="LPAREN" string="(" />
<token line="9" type="NUMBER" string="0" />
<token line="9" type="RPAREN" string=")" />
<token line="9" type="SEMI" string=";" />
<token line="10" type="RBRACE" string="}" />
<token line="11" type="RBRACE" string="}" />
<token line="12" type="VOID" string="void" />
<token line="12" type="ID" string="main" />
<token line="12" type="LPAREN" string="(" />
<token line="12" type="RPAREN" string=")" />
<token line="12" type="LBRACE" string="{" />
<token line="13" type="INT" string="int" />
<token line="13" type="ID" string="yylineno" />
<token line="13" type="SEMI" string=";" />
<token line="14" type="INT" string="int" />
<token line="14" type="ID" string="t" />
<token line="14" type="SEMI" string=";" />

```

```

<token line="15" type="INT" string="int" />
<token line="15" type="ID" string="yytext" />
<token line="15" type="SEMI" string=";" />
<token line="16" type="ID" string="t" />
<token line="16" type="ASSIGN" string="=" />
<token line="16" type="ID" string="foo" />
<token line="16" type="LPAREN" string="(" />
<token line="16" type="NUMBER" string="3" />
<token line="16" type="COMMA" string="," />
<token line="16" type="NUMBER" string="4" />
<token line="16" type="RPAREN" string=")" />
<token line="16" type="SEMI" string=";" />
<token line="17" type="ID" string="bar" />
<token line="17" type="LPAREN" string="(" />
<token line="17" type="ID" string="t" />
<token line="17" type="RPAREN" string=")" />
<token line="17" type="SEMI" string=";" />
<token line="18" type="RBRACE" string="}" />
</root>

```

c_0 语言文件 testlex.c

```

void f1(int a,int b) {
    a = 1;
    b = a+b;
}
void main()
{
    int a[100];
    int b;
    float c;
    a[b]=a;
    if(c<b){
        f1(a,b);
    }
}

```

c_0 语言文件 testlex.c的输出

```

<?xml version="1.0"?>
<root>
  <token line="1" type="VOID" string="void" />
  <token line="1" type="ID" string="f1" />
  <token line="1" type="LPAREN" string="(" />
  <token line="1" type="INT" string="int" />
  <token line="1" type="ID" string="a" />
  <token line="1" type="COMMA" string="," />
  <token line="1" type="INT" string="int" />
  <token line="1" type="ID" string="b" />
  <token line="1" type="RPAREN" string=")" />

```

```

<token line="1" type="LBRACE" string="{ " />
<token line="2" type="ID" string="a" />
<token line="2" type="ASSIGN" string="=" />
<token line="2" type="NUMBER" string="1" />
<token line="2" type="SEMI" string=";" />
<token line="3" type="ID" string="b" />
<token line="3" type="ASSIGN" string="=" />
<token line="3" type="ID" string="a" />
<token line="3" type="PLUS" string="+" />
<token line="3" type="ID" string="b" />
<token line="3" type="SEMI" string=";" />
<token line="4" type="RBRACE" string="}" />
<token line="5" type="VOID" string="void" />
<token line="5" type="ID" string="main" />
<token line="5" type="LPAREN" string="(" />
<token line="5" type="RPAREN" string=")" />
<token line="6" type="LBRACE" string="{ " />
<token line="7" type="INT" string="int" />
<token line="7" type="ID" string="a" />
<token line="7" type="LSQUAR" string="[" />
<token line="7" type="NUMBER" string="100" />
<token line="7" type="RSQUAR" string="]" />
<token line="7" type="SEMI" string=";" />
<token line="8" type="INT" string="int" />
<token line="8" type="ID" string="b" />
<token line="8" type="SEMI" string=";" />
<token line="9" type="FLOAT" string="float" />
<token line="9" type="ID" string="c" />
<token line="9" type="SEMI" string=";" />
<token line="10" type="ID" string="a" />
<token line="10" type="LSQUAR" string="[" />
<token line="10" type="ID" string="b" />
<token line="10" type="RSQUAR" string="]" />
<token line="10" type="ASSIGN" string="=" />
<token line="10" type="ID" string="a" />
<token line="10" type="SEMI" string=";" />
<token line="11" type="IF" string="if" />
<token line="11" type="LPAREN" string="(" />
<token line="11" type="ID" string="c" />
<token line="11" type="LT" string="<" />
<token line="11" type="ID" string="b" />
<token line="11" type="RPAREN" string=")" />
<token line="11" type="LBRACE" string="{ " />
<token line="12" type="ID" string="f1" />
<token line="12" type="LPAREN" string="(" />
<token line="12" type="ID" string="a" />
<token line="12" type="COMMA" string="," />
<token line="12" type="ID" string="b" />
<token line="12" type="RPAREN" string=")" />
<token line="12" type="SEMI" string=";" />

```

```
<token line="13" type="RBRACE" string="}" />
<token line="14" type="RBRACE" string="}" />
</root>
```

c₀ 语言文件 testparser.c

```
int a;
int b;
int d,e,f;

void main()
{
    int a=0;
    int b=2;
    while(a==b)
        ++a;
}
```

c₀ 语言文件 testparser.c的输出

```
<?xml version="1.0"?>
<root>
  <token line="1" type="INT" string="int" />
  <token line="1" type="ID" string="a" />
  <token line="1" type="SEMI" string=";" />
  <token line="2" type="INT" string="int" />
  <token line="2" type="ID" string="b" />
  <token line="2" type="SEMI" string=";" />
  <token line="3" type="INT" string="int" />
  <token line="3" type="ID" string="d" />
  <token line="3" type="COMMA" string="," />
  <token line="3" type="ID" string="e" />
  <token line="3" type="COMMA" string="," />
  <token line="3" type="ID" string="f" />
  <token line="3" type="SEMI" string=";" />
  <token line="5" type="VOID" string="void" />
  <token line="5" type="ID" string="main" />
  <token line="5" type="LPAREN" string="(" />
  <token line="5" type="RPAREN" string=")" />
  <token line="6" type="LBRACE" string="{" />
  <token line="7" type="INT" string="int" />
  <token line="7" type="ID" string="a" />
  <token line="7" type="ASSIGN" string="=" />
  <token line="7" type="NUMBER" string="0" />
  <token line="7" type="SEMI" string=";" />
  <token line="8" type="INT" string="int" />
  <token line="8" type="ID" string="b" />
  <token line="8" type="ASSIGN" string="=" />
  <token line="8" type="NUMBER" string="2" />
  <token line="8" type="SEMI" string=";" />
```

```
<token line="9" type="WHILE" string="while" />
<token line="9" type="LPAREN" string="(" />
<token line="9" type="ID" string="a" />
<token line="9" type="EQ" string="==" />
<token line="9" type="ID" string="b" />
<token line="9" type="RPAREN" string=")" />
<token line="10" type="INC_OP" string="++" />
<token line="10" type="ID" string="a" />
<token line="10" type="SEMI" string=";" />
<token line="11" type="RBRACE" string="}" />
</root>
```
