

中国海洋大学计算机科学与技术系

编译原理实验报告

姓名	岳宇轩	年级	2019	专业	计算机科学与技术
学号	19020011038	题目	Cygwin 环境的熟悉和 lex 的使用 1、2		
实验时间	2022 年 4 月 7-14 日			实验教师	王欣捷

一、实验目的：

熟悉 cygwin 环境的使用，学习使用 lex 写简单的词法分析程序，会在 cygwin 环境下使用 flex 调试 lex 写的程序。

二、实验环境：

Cygwin、GCC、Flex（或者你自己搭建的环境）

三、实验任务及要求：

实验一：

1、读懂 exam1.l 和 exam2.l 两个例子，使用 cygwin 下的 flex 工具将 exam1.l 和 exam2.l 编译并调试通过。并且修改 exam2.l，在其基础上增加如下记号：

- 左右大小括号：{ } ()
- 将关系算符改写成 C 中的形式
- 分号、赋值号：； =
- 关键字：if else
- 双斜线表示的注释：//
- 算术运算符号：+ - * /
- 将标识符改为可含有下划线，并且可以以下划线开头
- 将注释内容忽略

2、尝试实验内容：完善 exam1.l 中 installID 和 installNum 两辅助函数的功能

3、实验要求：在 cygwin 下用 flex 和 gcc 工具将实验调试通过，并写出测试例测试正确性。

4、实验参考：exam1.l 和 exam2.l。请认真阅读例子，发现错误及时提出。Test1.p 和 test2.p 可分别作为两个程序的测试用例。

实验二：

在实验 1 所改写的程序的基础上增加识别 `string` 记号。`string` 是字符串，如果”出现在字符串中，则必须转义，写成`\`形式；如果`\`出现在字符串中，也必须转义，写成`\\`形式。

实验要求：在 `cygwin` 下用 `flex` 和 `gcc` 工具将实验调试通过，并写出测试例测试正确性。同时该实验必须满足如下要求：

1. `string` 是字符串，它是以双引号括起的一串字符。
2. 双引号内的字符有如下要求：
 - 不能包含单独的”或者`\`，除非用`\`进行转义。例如字符串内的”写成`\`”，而`\`写成`\\`。
 - 字符串内可以出现转义字符。（例如：`\n`,`\t`,`\"`,`\\`,`^\c`,`\ddd`，其中 `c` 表示任意可打印字符，`d` 表示数字。）此条可简化为字符串内可包含以`\`开头的任意字符。
 - 字符串内不可包含实体的换行。（可以包含`\n`，但是如果两个“”中的字符串出现在两行中，即包含了实体换行，则不应识别为字符串）

测试用例：使用 `test2-1.i` 和 `test2-1.o` 来验证你的代码的正确性，`.i` 为输入文件，`.o` 为输出，请对比你的代码的输出和`.o` 文件的输出是否逻辑上一致。

四、实验过程及结果：

1、实验过程

实验一

1) 读懂 `exam1.1` 和 `exam2.1` 两个例子，使用 `cygwin` 下的 `flex` 工具将 `exam1.1` 和 `exam2.1` 编译并调试通过。

2) 并且修改 `exam2.1`，在其基础上增加如下记号：

➤ 左右大小括号：`{ } ()`

首先加入记号 `KUOHAO`：

```
#define KUOHAO
```

7

在翻译规则中加入：

```
<INITIAL> "("|"{"|")"|"}"          {return (KUOHAO);}
```

最后在 `writout` 函数中加入：

```
case KUOHAO:fprintf(yyout,"(KUOHAO,\"%s\")",yytext);break;
```

➤ 将关系运算符改写成 C 中的形式

```
<INITIAL> "<"| "<="| "=="| "!="| ">"| ">=" {return (RELOP);}
```

➤ 分号、赋值号：； =

添加记号

```
#define FENHAO 8
#define FUZHI 9
```

在翻译规则中加入

```
<INITIAL> ";" {return (FENHAO);}
<INITIAL> "=" {return (FUZHI);}
```

最后在 writeout 函数中加入

```
case FENHAO:fprintf(yyout,"(FENHAO,\"%s\")",yytext);break;
case FUZHI:fprintf(yyout,"(FUZHI,\"%s\")",yytext);break;
```

➤ 关键字：if else

加入记号

```
#define IF 10
#define ELSE 11
```

在翻译规则中加入

```
<INITIAL> "if" {return (IF);}
<INITIAL> "else" {return (ELSE);}
```

最后，在 writeout 函数中加入

```
case IF:fprintf(yyout,"(IF,\"%s\")",yytext);break;
case ELSE:fprintf(yyout,"(ELSE,\"%s\")",yytext);break;
```

➤ 双斜线表示的注释：//

增加一个状态 ZHUSHI 表示单行的注释

%s ZHUSHI

状态转换如下：

- 在初始状态 INITIAL 时遇到//则转入单行注释状态 ZHUSHI
- 在单行注释状态 ZHUSHI 时遇到换行符\n 表示一行注释的结束，则转入初始状态 INITIAL
- 在 ZHUSHI 状态遇到其它字符不操作

```
<INITIAL> "/"                                {BEGIN ZHUSHI;}
<ZHUSHI> \n                                    {BEGIN INITIAL;}
<ZHUSHI> .                                     {}
```

➤ 算术运算符: + - * /

定义记号算术运算符 SSYSF

```
#define SSYSF
```

12

翻译规则中加入:

```
<INITIAL> "+"|"-"|"*"|"|" "/"                {return (SSYSF);}
```

最后在 writeout 中加入:

```
case SSYSF:fprintf(yyout, "(SSYSF,\"%s\")",yytext);break;
```

➤ 将标识符改为可含有下划线, 并且可以以下划线开头

只需在正规定义的 letter 中加入下划线_即可, 这样标识符就可含有下划线

```
letter    [_A-Za-z]
```

根据 id 的定义, id 第一个为 letter 中元素, 而 letter 中含有下划线, 因此标识符可以以下划线开头

```
id        {letter}({letter}|{digit})*
```

➤ 将注释内容忽略

将 ECHO;去掉即可, 相当于不进行输出。

```
<INITIAL> "/*"                                {BEGIN COMMENT;}
<COMMENT> "*/"                                {BEGIN INITIAL;}
<COMMENT> .\n                                  {}
```

实验二

1.定义记号 STRING

#define STRING

13

2.添加正规式

```
zhuanyi  (\\\\|\\\"|\\'|\\\\{letter})|\\\\{digit})
string    \"({zhuanyi})[\\\"'\\\\n\\\\\\\\]*\"
```

首先定义转移字符，根据实验要求：

- 不能包含单独的”或者\，除非用\进行转义。例如字符串内的”写成\”，而\写成\\。
- 字符串内可以出现转义字符。（例如：\n,\t,\\,\\^c, \ddd, 其中 c 表示任意可打印字符, d 表示数字。）
此条可简化为字符串内可包含以\开头的任意字符。

故转义字符的集合为\\ \” \’ \letter \digit 组成

由于需要使用\转义，而转义字符\也要需要转移，故在上述集合需要转义出要加两个\

其次定义字符串，字符串内可以出现转义字符，用 zhuanyi 表示即可；除了”‘\n 以外的字符需要用^将其去除。字符串以”开头，以”结尾，”需要转义，故 string 的开头结尾是\”

另外，考虑到空串的情况，可以用()*表示

3.在翻译规则中添加

```
<INITIAL>{string}                                {return (STRING);}
```

4.在 writeout 中添加

```
case STRING:fprintf(yyout, "(STRING,%s)", yytext);break;
```

2、实验结果

1)

```
~/example
yyx@LAPTOP-2ND02015 ~/example
$ ls
exam1.l  exam1test.p  exam2.l  exam2test.p

yyx@LAPTOP-2ND02015 ~/example
$ flex exam1.l

yyx@LAPTOP-2ND02015 ~/example
$ ls
exam1.l  exam1test.p  exam2.l  exam2test.p  lex.yy.c

yyx@LAPTOP-2ND02015 ~/example
$ gcc lex.yy.c -lf1

yyx@LAPTOP-2ND02015 ~/example
$ ls
a.exe  exam1.l  exam1test.p  exam2.l  exam2test.p  lex.yy.c

yyx@LAPTOP-2ND02015 ~/example
$ ./a.exe exam1test.p
(WHILE, "while") (ID, "a") (RELOP, ">=") (ERRORCHAR, "-") (NUM, "1.2E-2")
(DO, "do") (ID, "b") (RELOP, "<=") (NUM, "2")
yyx@LAPTOP-2ND02015 ~/example
$ |
```

编译并调试通过 exam1.l

2)

```
~/example
a.exe  exam1.l  exam1test.p  exam2.l  exam2test.p  lex.yy.c

yyx@LAPTOP-2ND02015 ~/example
$ flex exam2.l

yyx@LAPTOP-2ND02015 ~/example
$ ls
a.exe  exam1.l  exam1test.p  exam2.l  exam2test.p  lex.yy.c

yyx@LAPTOP-2ND02015 ~/example
$ gcc lex.yy.c -lf1

yyx@LAPTOP-2ND02015 ~/example
$ ls
a.exe  exam1.l  exam1test.p  exam2.l  exam2test.p  lex.yy.c

yyx@LAPTOP-2ND02015 ~/example
$ ./a.exe exam2test.p
(WHILE, "while") (ID, "a") (RELOP, ">=") (ERRORCHAR, "-") (NUM, "1.2E-2")
(DO, "do") (ID, "b") (RELOP, "<=") (NUM, "2") /* 请注意：测试文件的格式必须符合
要求。
  比如，该文件要求的格式是UNIX格式。*/
yyx@LAPTOP-2ND02015 ~/example
$
```

编译并调试通过 exam2.l

3)


```
~/example
yyx@LAPTOP-2ND02015 ~/example
$ gcc lex.yy.c -lf1

yyx@LAPTOP-2ND02015 ~/example
$ ./a.exe test1.i
(IF,"if")(KUOHAO,"(")(ID, "_x") (RELOP, "==") (NUM, "1.2E-2")
(KUOHAO,")") (KUOHAO,"{")(ID, "x") (FUZHI, "=") (NUM, "1")
(SSYSF, "+") (NUM, "2") (SSYSF, "-") (NUM, "3") (SSYSF, "*")
(NUM, "4") (SSYSF, "/" ) (NUM, "5") (FENHAO, ";") (KUOHAO, "}")
(ELSE, "else")(IF,"if")(KUOHAO,"(")(ID, "_x") (RELOP, "!=")
(NUM, "1.2") (KUOHAO,")") (KUOHAO,"{")(ID, "x_1") (FUZHI, "=")
(NUM, "1") (FENHAO, ";") (KUOHAO, "}") (WHILE, "while") (KUOHAO,"(")
(ID, "a_b") (RELOP, ">=") (NUM, "1.2E-2") (KUOHAO,"") (DO, "do")
(ID, "_b") (RELOP, "<") (NUM, "2") (FENHAO, ";") (NUM, "1")
(ERRORCHAR, ".") (ERRORCHAR, ".") (NUM, "2")
yyx@LAPTOP-2ND02015 ~/example
$
```

修改 exam2.l 之后输入 test1.i 得到的输出结果，经过与 test1.o 的比对结果相同
(对标记的命名及合并略有不同，但识别准确)

4)

```
yyx@LAPTOP-2ND02015 ~/example
$ ./a.exe test2-1.i
(STRING,"string1")(STRING,"string2\\")(STRING,"string3\\")(STRING,"string4\n\\0a")(ERRORCHAR, "")
(ID, "string5") (ERRORCHAR, "\\") (ERRORCHAR, "") (STRING,"string6")(ERRORCHAR, "")
(ERRORCHAR, "") (ID, "string7") (ERRORCHAR, "") (ERRORCHAR, "") (ID, "string8")
(ERRORCHAR, "") (ERRORCHAR, "")
yyx@LAPTOP-2ND02015 ~/example
$
```

实验 2 的结果：与 test2-1.o 结果相同

五、实验总结

- 1、加入新的标记总共有三步骤：定义标记、添加翻译规则、在 writeout 函数中输出
- 2、可以通过 %s 定义新的状态，同样也可以在状态之间进行转换
- 3、去掉 ECHO;即不进行输出
- 4、熟悉了正规式的定义和使用
- 5、在实验二添加识别字符串时，我更加清楚了转义字符的使用，同时也更加熟练的掌握了如何定义正规式。在做这个实验的时候，我遇到一个问题，就是我识别的都正确，但是最终结果都会多出一对”，通过询问老师的时候，我得知我是把”也作为了字符串的一部分了。如果仅在 writeout 函数中去掉输出格式 %s 两侧的”也能得到和 test2-1.o 相同的结果，但”的含义是不同的，一个是输出格式中的”，一个是识别字符串中的”。应当在返回结果时就去掉”，因为字符串的内容应该是不包含”的。