Lex 词法分析器分析 string

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、实验内容：

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。

三、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。）

四、具体实现

对 exam2.1 文件修改如下：

1. 新增声明：

```
26
27  #define STRING      29
```

2. 新增对字符串的正规定义：

```
31  delim       [ \t \n]
32  ws          {delim}+
33  letter_     [A-Za-z_]
34  digit       [0-9]
35  id          {letter_}({letter_}|{digit})*
36  number      {digit}+(\.{digit}+)?(E[+-]?{digit}+)?
37  string      \"(\\.|[^\n\\\"])*\"
```

对正规定义的解释：string　　\"(\\.|[^\n\\\"])*\"

- \"　　对引号"的转义，字符串以"开头和结尾。
- \\.　　在 Lex 源程序中词法规则中规定点.匹配除换行之外的任何字符，所以该部分表示满足要求：字符串内可包含以\开头的任意字符。
- [^\n\\\"] ^表示补集，[^…]表示补集，即匹配除^之后所列字符以外的任何字符。所以该部分表示满足要求：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\\。同时满足字符串的正常字符要求。

3. 新增对 string 的翻译规则：

```
<INITIAL>{ws}           {;}
<INITIAL>while          {return (WHILE);}
<INITIAL>do             {return (DO);}
<INITIAL>if             {return (IF);}
<INITIAL>else           {return (ELSE);}
<INITIAL>{id}           {return (ID);}
<INITIAL>{number}       {return (NUMBER);}
<INITIAL>{string}       {return (STRING);}
```

4. 输出函数新增对 string 的输出：

```
void writeout(int c){
  switch(c){
    case ERRORCHAR: fprintf(yyout, "(ERRORCHAR, \"%s\") ", yytext);break;
    case RELOP: fprintf(yyout, "(RELOP, \"%s\") ", yytext);break;
    case WHILE: fprintf(yyout, "(WHILE, \"%s\") ", yytext);break;
    case DO: fprintf(yyout, "(DO, \"%s\") ", yytext);break;
    case IF: fprintf(yyout, "(IF, \"%s\") ", yytext);break;
    case ELSE: fprintf(yyout, "(ELSE, \"%s\") ", yytext);break;
    case NUMBER: fprintf(yyout, "(NUM, \"%s\") ", yytext);break;
    case ID: fprintf(yyout, "(ID, \"%s\") ", yytext);break;
    case NEWLINE: fprintf(yyout, "\n");break;
    case COPY: fprintf(yyout, "(COPY, \"%s\") ", yytext);break;
    case SEMICOLON: fprintf(yyout, "(SEMICOLON, \"%s\") ", yytext);break;
    case BRACKET: fprintf(yyout, "(BRACKET, \"%s\") ", yytext);break;
    case OP: fprintf(yyout, "(OP, \"%s\") ", yytext);break;
    case STRING: fprintf(yyout, "(STRING, \"%s\") ", yytext);break;
    default:break;
  }
  return;
}
```

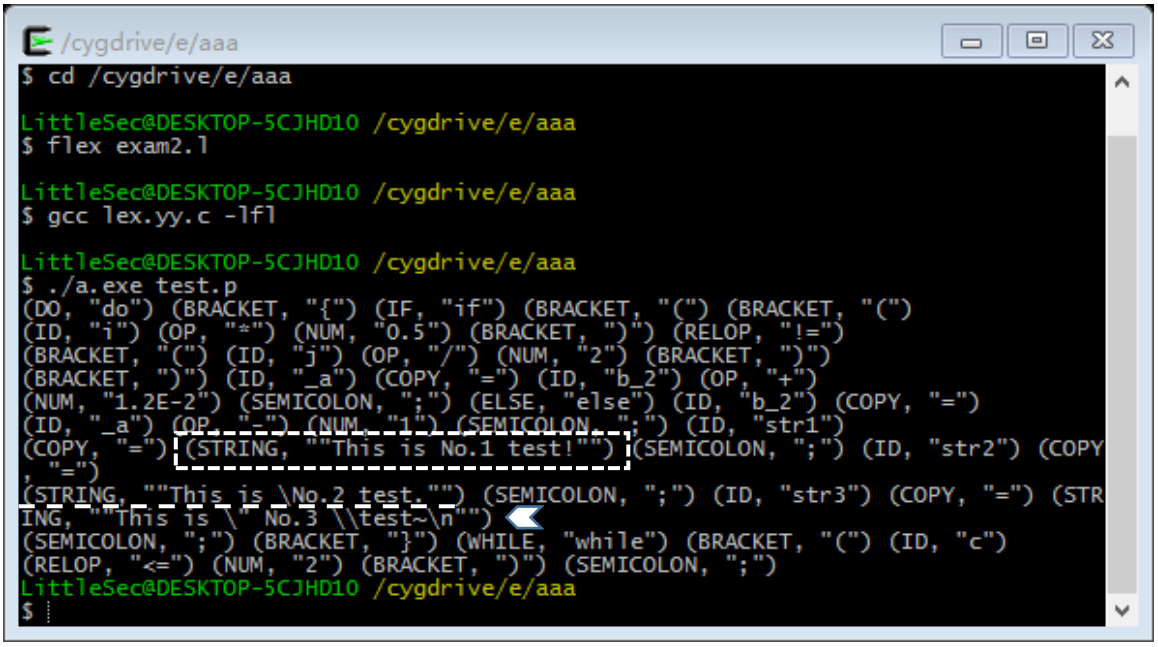
5. 测试文件 test.p 新增内容如下：

```

//这是第一条注释，双斜杠的

do {
    if((i * 0.5) != (j / 2))
        _a = b_2 + 1.2E-2;
    else
        b_2 = _a - 1;
    str1 = "This is No.1 test!";
    str2 = "This is \No.2 test.";
    str3 = "This is \" No.3 \\test~\n";
}while (c <= 2);

/*
这是
第二条注释
斜杠星号的
*/
```

测试文件中包含：

- ✓ 原有的测试内容。
- ✓ str1 是正常的字符串。
- ✓ str2 是含有以\开头的正常字符。
- ✓ str3 是含有对引号"、反斜杠\、回车\n 对转义。

6. 在 cygwin 下用 flex 编译器对新的 test2.l 进行编译，生成 lex.yy.c 文件，用 gcc 编译器对该 C 源程序进行编译得到 a.exe 文件，用测试文件 test.p 进行测试，如图所示：



经核对，结果正确。

五、 心得与体会

1. 进一步熟悉 flex 词法分析器生成工具的使用，Lex 的语法规则和组织方式。
2. 写正规定义前应当仔细阅读 Lex 使用手册，了解 Lex 的语法规则，例如 Lex 对点和^的规定，这样会方便自己写出正规定义。
3. 由于字符串以"开始，以"结束，且中间可以包含转义字符\"，所以不能简单的将两个引号之间的部分识别为字符串，因为最后一个引号可能是\"的形式，这将构成一个词法错误。
4. 实际上设计正规定义时，只需要考虑特殊情况，通过 Lex 语法规则中的补集这个概念对他们求补集即可匹配除特殊情况外的情况，点和补集就是很好的语法规则。
5. 可以加上对错误情况的处理设计，例如字符串中直接出现\或等需要转义的字符。