# Compilers Principals - Lab1

**Zhixin Zhang, 3210106357**

## 1 实验内容

本次实验，我们利用 Flex 和 Bison 实现了 sysY 语言的词法分析和语法分析． 通过

```
make compiler
./compiler <input file>
```

可以对输入的 sy 文件进行语法检查， 如果可以正确解析出语法树，程序将正常退出并返回 0．同时在错误流中显示

```
Parse success!
```

否则，程序将汇报错误，一个错误的代码的解析输出如下：

```
Error at Line 5 Mysterious character"@".
error: syntax error
Failed to parse the file : tests/lab1/error1.sy
```

## 2 代码实现

### 2.1 主接口

```
int main(int argc, char **argv)
{
    yylineno = 1;
    if (argc < 2)
    {
    std::cerr << "Usage: " << argv[0] << "<input file> [output file]" << std::endl;
        return 1;
    }
  if(!(yyin = fopen(argv[1], "r")))
  {
    std::cerr << "Open file error : " << argv[1] << std::endl;
    return 1;
  }
  if(yyparse())
  {
    std::cerr << "Failed to parse the file : " << argv[1] << std::endl;
    return 1;
  }
  // if(!line_error) // 尚未完整实现
  //  Print_Tree(Root, 0);
  std::cerr << "\nParse success !" << std::endl;
    fclose(yyin);
    return 0;
}
```

## 2.2 Flex 词法分析

报错方法：

```
void error_print(int line, std::string text, std::string msg)
{
    if (line_error == line) return;
    line_error = line;
    std::cout << "Error at Line " << line << " " << msg << "\"" << text << "\".\n";
}
```

正则表达式定义：

```
digit [0-9]
blank [ \t\r\n]
letter [a-zA-Z]
alpha _|{letter}
Comment1 "/*"[^*]*"*"+([^*/][^*]*"*"+)*"/"
Comment2 "//"[^\r\n]*
ident {alpha}({alpha}|{digit})*
oct 0[0-7]+
hex 0[Xx][0-9a-fA-F]+
zero 0
n_zero [1-9]+{digit}*
integer {zero}|{n_zero}
newline "\n"
ws [ \r\t\n]+
```

词法解析部分：

```
{Comment1}      {  /* nothing to do */ }
{Comment2}      {   /* nothing to do */ }
"int"           {   return INT; }
"void"          {   return VOID; }
"if"            {   return IF; }
"else"          {   return ELSE; }
"while"         {   return WHILE; }
"+"             {   return ADD; }
// ..............................
{ident}         {   return IDENT; }
{hex}           {   return INTCONST; }
{oct}           {   return INTCONST; }
{integer}       {   return INTCONST; }
{ws}            {   /* nothing to do */}
.               {
                    error_print(yylineno, std::string(yytext), "Mysterious character");
                    return 0;
                }
```

## 2.3 Bison 语法分析

从 Root 开始进行自底向上分析.

部分语法分析如下： 需要适当修改语法，避免二义性.

```
ROOT        : CompUnit
            ;
CompUnit    : Decl
            | FuncDef
            | CompUnit Decl
            | CompUnit FuncDef
            ;
Decl        : VarDecl
            ;
BType       : INT
            ;
VarDef      : IDENT
            | IDENT Widths
            | IDENT ASSIGN InitVal
            | IDENT Widths ASSIGN InitVal
            ;
VarDecl     : BType VarDef VarDefs SEMI
            ;
VarDefs     : VarDefs COMMA VarDef
            |
            ;
Widths      : "[" INTCONST "]"
            | Widths "[" INTCONST "]"
            ;
InitVal     : Exp
            | "{"  "}"
            | "{" InitVals "}"
            ;
InitVals    : InitVal
            | InitVals COMMA InitVal
            ;
FuncHead    : BType IDENT
            | VOID IDENT
            ;
FuncDef     : FuncHead "(" FuncParams ")" Block
            | FuncHead "(" ")" Block
            ;
```

## 3 测试结果

```
python3 test.py ./compiler lab1
```

tests 下的测试样例全部通过：


图 1 All tests passed!