中国矿业大学计算机学院

系统软件开发实践报告

**课程名称** 系统软件开发实践

**报告时间** 2020年4月22日

**学生姓名** 陆玺文

**学 号** 03170908

**专 业** 计算机科学与技术

**任课教师** 张博

成绩考核

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 课程教学目标 | | | 占比 | | 得分 |
| 1 | **目标1：**针对编译器中词法分析器软件要求，能够分析系统需求，并采用FLEX脚本语言描述单词结构。 | | | 15% | |  |
| 2 | **目标2：**针对编译器中语法分析器软件要求，能够分析系统需求，并采用Bison脚本语言描述语法结构。 | | | 15% | |  |
| 3 | **目标3：**针对计算器需求描述，采用Flex/Bison设计实现高级解释器，进行系统设计，形成结构化设计方案。 | | | 30% | |  |
| 4 | **目标4：**针对编译器软件前端与后端的需求描述，采用软件工程进行系统分析、设计和实现，形成工程方案。 | | | 30% | |  |
| 5 | **目标5：**培养独立解决问题的能力,理解并遵守计算机职业道德和规范，具有良好的法律意识、社会公德和社会责任感。 | | | 10% | |  |
| 总成绩 | | | | | |  |
| 指导教师 | |  | 评阅日期 | |  | |

目 录

[1、 实验三 Bison实验一 1](#_Toc38961620)

[1.1 实验目的 1](#_Toc38961621)

[1.2 实验内容 1](#_Toc38961622)

[1.3 环境配置与使用 1](#_Toc38961623)

[1.3.1 Windows环境下 1](#_Toc38961624)

[1.3.2 CentOS环境下 1](#_Toc38961625)

[1.4 源代码分析 2](#_Toc38961626)

[1.4.1 Flex代码分析 2](#_Toc38961627)

[1.4.2 Bison代码分析 3](#_Toc38961628)

[1.4.3 Bison的语法规则 4](#_Toc38961629)

[1.4.4 语法分析树 4](#_Toc38961630)

[1.5 实验结果 5](#_Toc38961631)

[1.5.1 Windows下实验结果 5](#_Toc38961632)

[1.5.2 CentOS下运行结果 5](#_Toc38961633)

[1.6 实验思考 5](#_Toc38961634)

[1.6.1 Warning1: incompatible implicit declaration of built-in function 'strdup' 6](#_Toc38961635)

[1.6.2 Warning2: assignment makes integer from pointer without a cast 6](#_Toc38961636)

[1.7 实验收获 6](#_Toc38961637)

# 实验三 Bison实验1

## 实验目的

熟悉Flex与Bison协同工作，编写语法分析程序，测试Name.txt输出。

## 实验内容

1.阅读《Flex/Bison.pdf》第一、三章，掌握Bison基础知识。

2.利用Bison设计一个简单的语法分析器，掌握移进/规约分析，掌握语法分析树，掌握抽象语法树。

## 环境配置与使用

### Windows环境下

在前期实验的基础上，Windows下继续使用集成开发环境，通过点击不同的执行按钮来进行相应的操作如图 1‑1所示。

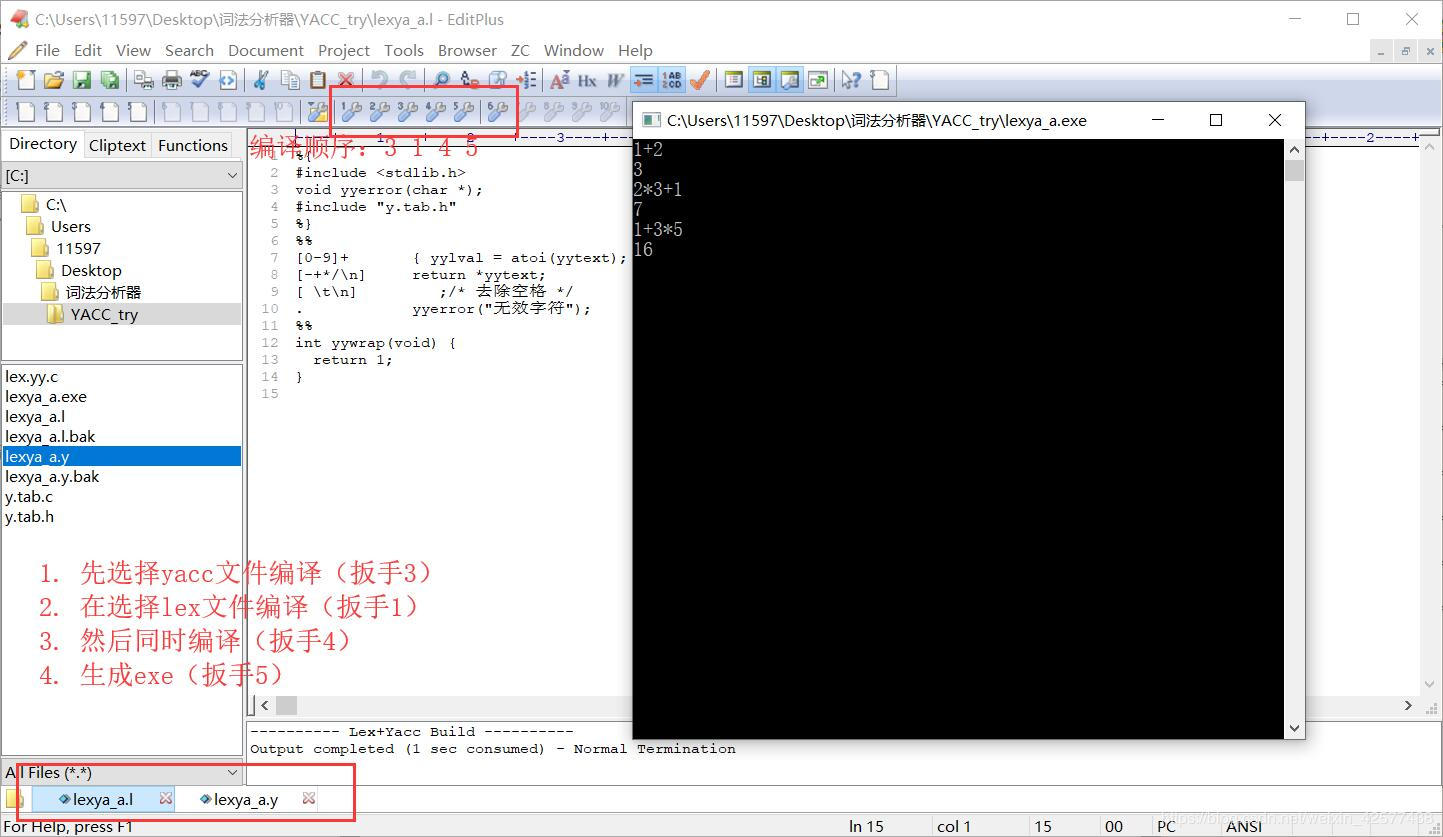


图 ‑1 Windows下集成环境使用图

### CentOS环境下

CentOS为腾讯云服务器，使用Xftp传输文件，Xshell远程连接终端来操作。

安装Bison使用命令“yum install bison”即可；编译yacc文件时，使用命令“bison -d bison1.y”同时生成bison1.tab.h与bison1.tab.c文件；链接生成c文件时，使用命令“cc -o parser bison1.tab.c biosn1.yy.c”即可，如图 1‑2所示。

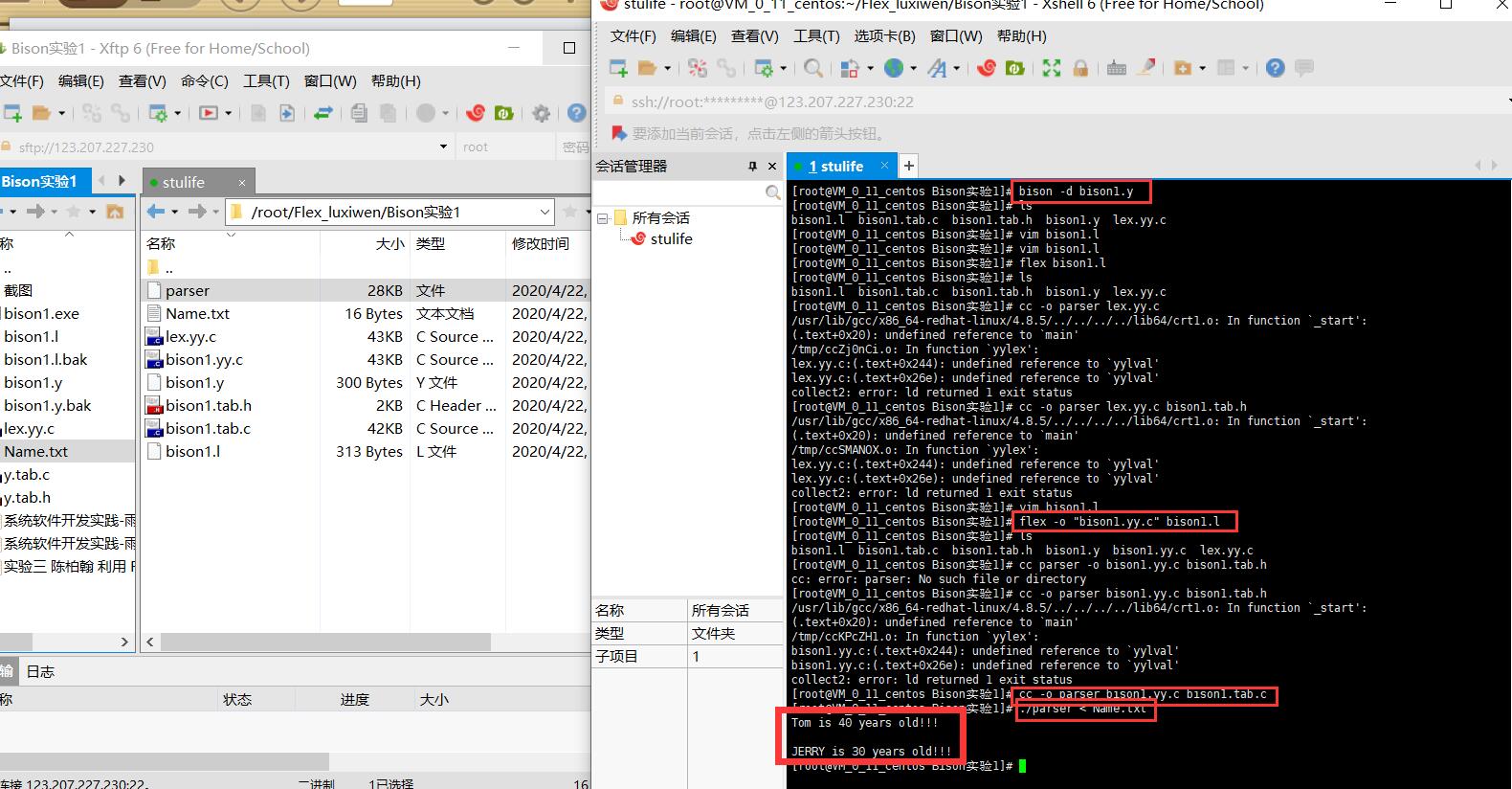


图 ‑2 CentOS下运行示意图

## 源代码分析

### Flex代码分析

Bison1.l整体上依据所发布的参考代码编写而成，针对部分提示的warning进行了一些相应的修改，详细代码如代码 1‑1所示。

代码 1‑1 bison1.l

|  |  |
| --- | --- |
|  | **%{** |
|  | #define YYSTYPE char\* |
|  | #include"y.tab.h" |
|  | #include <string.h> |
|  | **%}** |
|  | **char** **[**A**-**Za**-**z**]** |
|  | num **[**0**-**9**]** |
|  | eq **[=]** |
|  | name **{char}+** |
|  | age **{**num**}+** |
|  |  |
|  | **%%** |
|  | **{**name**}** **{**yylval **=** **(char\*)**strdup**(**yytext**);** **return** NAME**;}** |
|  | **{**eq**}** **{return** EQ**;}** |
|  | **{**age**}** **{**yylval **=** **(char\*)**strdup**(**yytext**);return** AGE**;}** |
|  |  |
|  | **%%** |
|  | **int** yywrap**(){return** 1**;}** |

行2，因为bison内部yylval的类型为YYSTYPE，将YYSTYPE定义为char\*有助于传递所有字符串。

行3，该.h头文件为bison编译之后生成的头文件，用于协作。

行4，所引入的头文件为strdup()函数所必须头文件。

行6~10，模式匹配规则。

行13~15，yylval为了向符号表中传递响应标识符的属性值，同时return标识符。

行18，yywrap指明分析器到达文件末尾时的下一步操作，yywrap返回1，词法分析器将返回一个零记号来表明文件结束。

### Bison代码分析

Bison1.y整体上依据所发布的参考代码编写而成，针对部分提示的warning进行了一些相应的修改，详细代码如代码 1‑2所示。

代码 ‑2 bison1.y

|  |  |
| --- | --- |
|  | **%{** |
|  | #include<stdio.h> |
|  | **%}** |
|  | **%**token NAME EQ AGE |
|  |  |
|  | **%%** |
|  | file **:**record |
|  | **|**record file |
|  | **;** |
|  |  |
|  | record **:**NAME EQ AGE **{**printf**("%s is %s years old!!!\n",**$1**,**$3**);}** |
|  | **;** |
|  |  |
|  | **%%** |
|  | **int** yyerror**(char** **\***msg**)** |
|  | **{** |
|  | printf**("Error encountered: %s \n",**msg**);** |
|  | **return** 0**;** |
|  | **}** |
|  | **int** main**()** |
|  | **{** |
|  | yyparse**();** |
|  | **return** 0**;** |
|  | **}** |

行4，token声明了定义的终结符。

行7~9，指明了文法规则file->record | record file | ε。

行11，指明了文法规则record -> NAME EQ AGE | ε。

行20，主程序中调用yyparse()开始语法分析。

### Bison的语法规则

Yacc是一个LALR（1）分析器自动生成器。Yacc与Lex一样，是贝尔实验室在UNIX上首先实现的，而且与Lex有直接的接口。Yacc的功能是，为2型文法自动生成基于LALR（1）的方法的语法语义分析器。按照LR分析应用于二义性文法的思想，即对二义性文法施加某些限定，Yacc同样可以适用于二义性文法分析器的自动生成（例如规定优先级和结合规则）。LALR（1）分析法是对LR（1）分析法的一种简化和改进，它的思想是对LR（1）中能够合并的项目集合并，从而减少状态。

Bison的规则基本上就是BNF（上下文无关文法），做了一点点简化以便易于输入。

### 语法分析树



图 ‑3 语法分析树（NAME=AGE）

在1.4.3中提到，bison使用的是LALR（1）分析方法，故以本例“NAME=AGE”为例，在从左向右识别过程中，首先会依次移进标识符NAME、EQ、AGE，接着根据语法record -> NAME EQ AGE，规约为record，然后根据file -> record，规约为file。

## 实验结果

在Windows与CentOS的环境下，程序均得到了正确的运行输出。

### Windows下实验结果

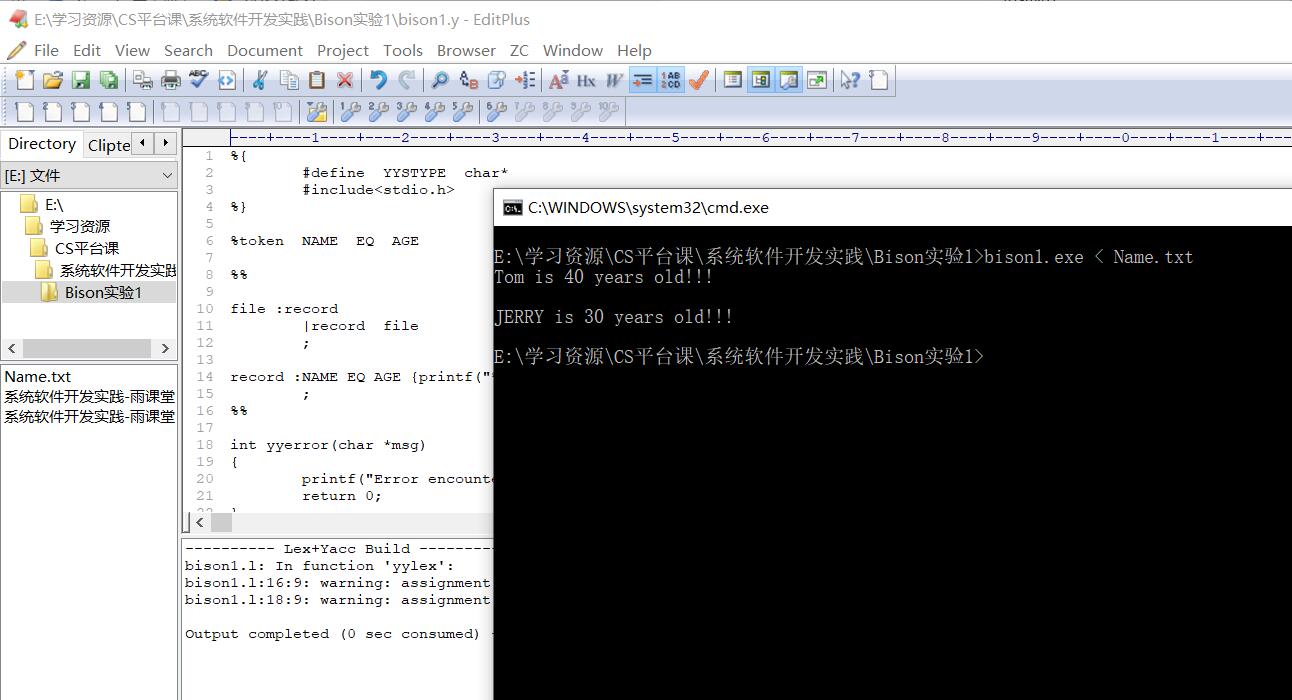


图 ‑4 Win下程序运行结果

### CentOS下运行结果

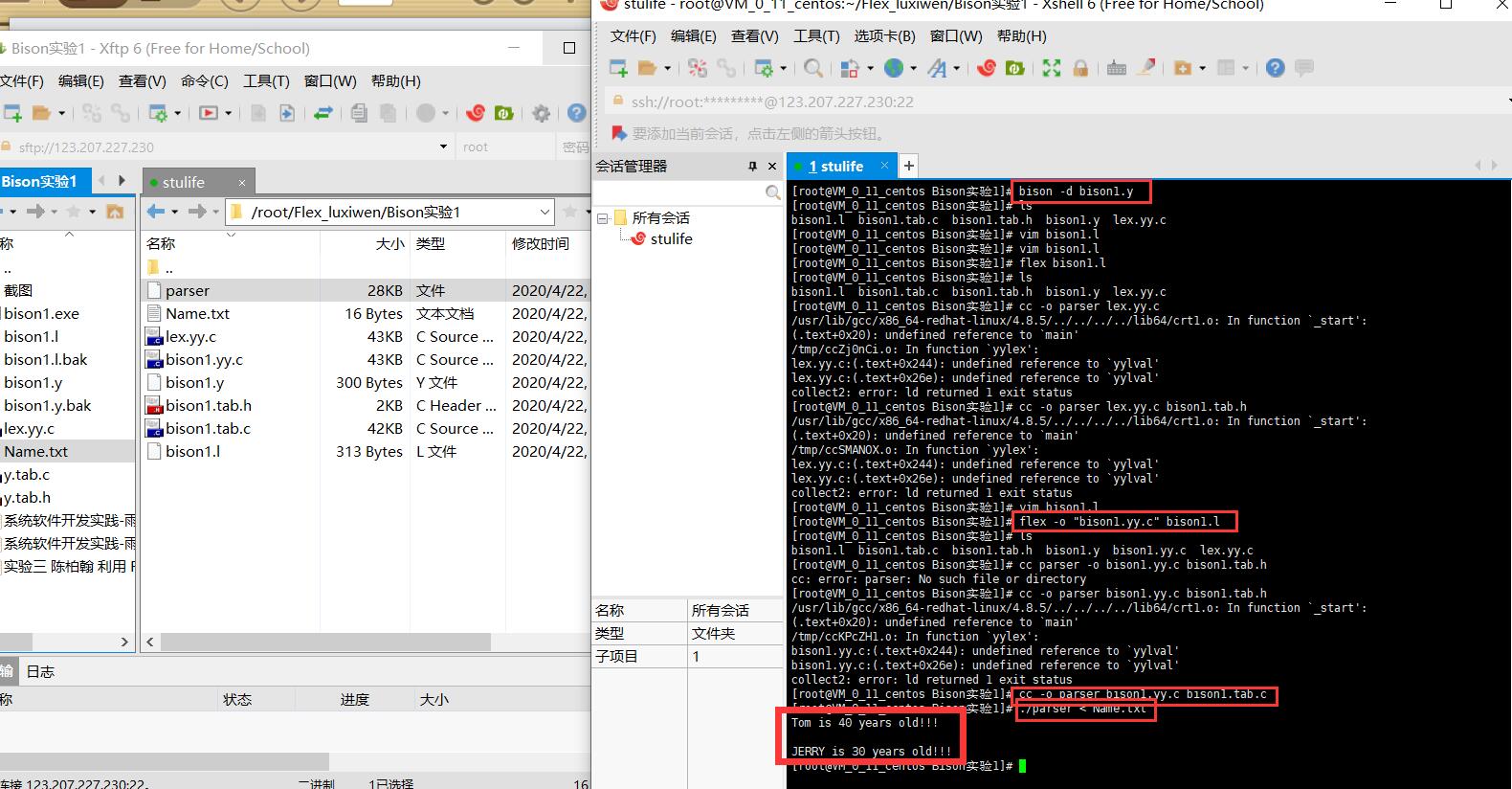


图 ‑5 CentOS下程序运行结果

## 实验思考

这次实验过程中，一开始遇到了两条警告错误，均关于语句“yylval = strdup(yytext)”，在查阅相关资料后找到了如下的解决方案。

### Warning1: incompatible implicit declaration of built-in function 'strdup'

这个错误的原因在于使用了strdup()，但是在之前并没有去进行相应的声明，在查阅资料后，可以通过导入头文件<string.h>来进行相应的解决。

参考资料：<https://zh.cppreference.com/w/c/experimental/dynamic/strdup> 。

### Warning2: assignment makes integer from pointer without a cast

这个错误的原因在于，strdup(yytext)返回值的类型为char\*，而在链接程序时，默认的yylval的类型为整型，因而出现了赋值时类型不匹配的错误。可以通过在.l文件头部加入#define yystype char\*来解决。

参考资料：<https://blog.csdn.net/backgarden_straw/article/details/7987665> 。

## 实验收获

这一次实验熟悉了使用Flex和yacc联合进行语法分析的步骤，对于一款编译器的诞生有了更加进一步的感受。在解决问题的过程中，对于yacc的自动生成有了更加进一步的认识。