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 1

1.1 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。

1.2 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。

1.3 环境配置与使用

1.3.1 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。

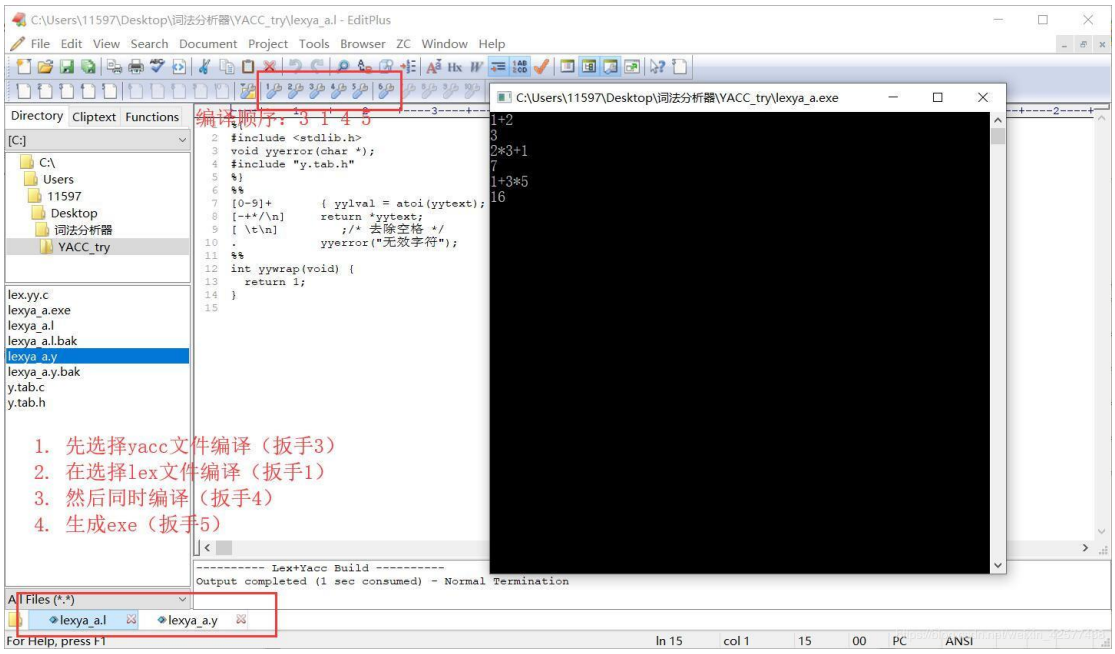

图 1-1　Windows 下集成环境使用图

1.3.2 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。

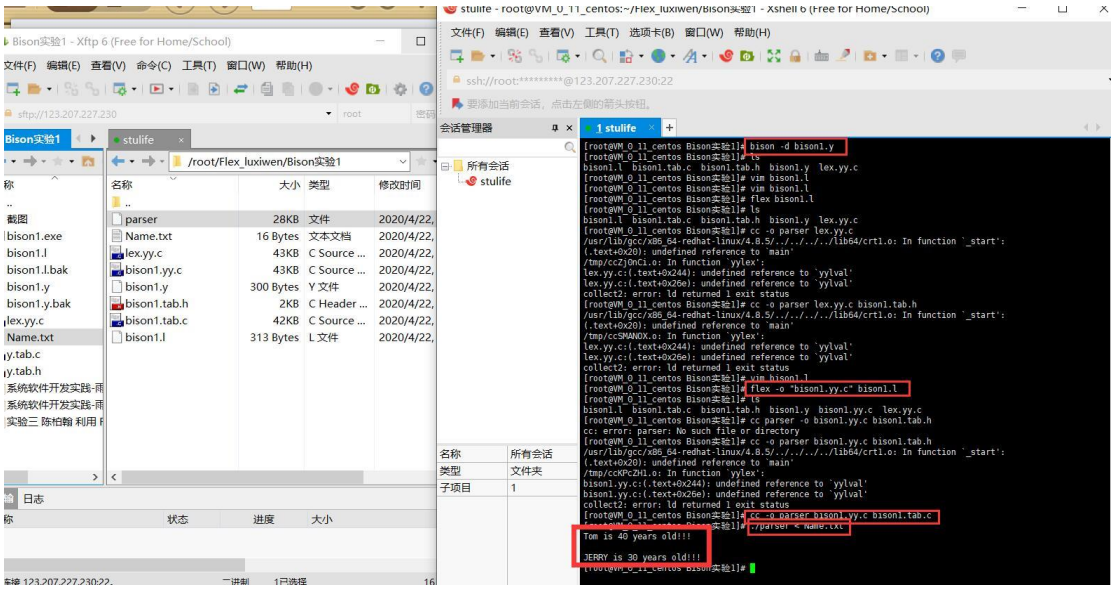

图 1-2 CentOS 下运行示意图

1.4 源代码分析

1.4.1 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。

1.4.2 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 1-2 bison1.y

```
1.  %{
2.      #include<stdio.h>
3.  %}
4.  %token NAME  EQ  AGE
5.
6.  %%
7.  file :record
8.      |record  file
9.      ;
10.
11. record :NAME EQ AGE {printf("%s is %s years
    old!!!\n",$1,$3);}
12.     ;
13.
14. %%
15. int yyerror(char *msg)
16. {
17.     printf("Error encountered: %s \n",msg);
18.     return 0;
19. }
20. int main()
21. {
22.     yyparse();
23.     return 0;
24. }
```

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。

1.4.3 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。

1.4.4 语法分析树

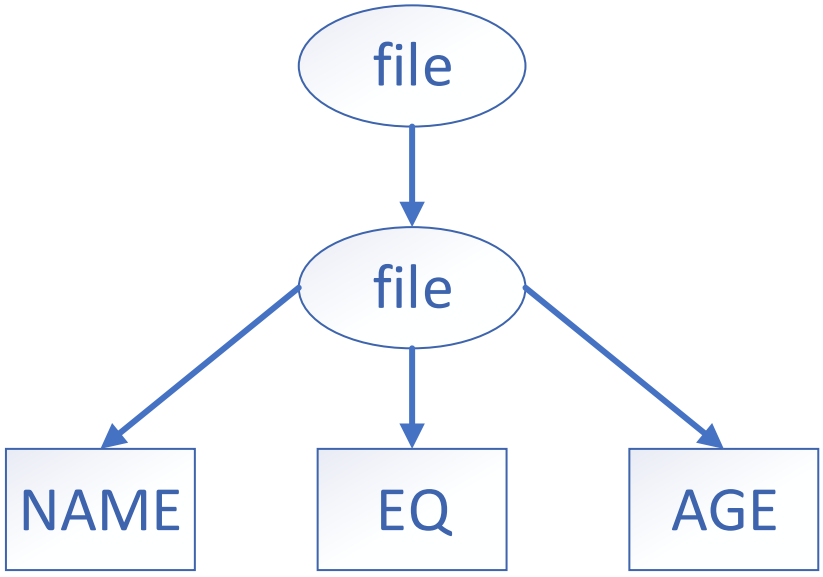

图 1-3　语法分析树（NAME=AGE）

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。

1.5 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。

1.5.1 Windows 下实验结果

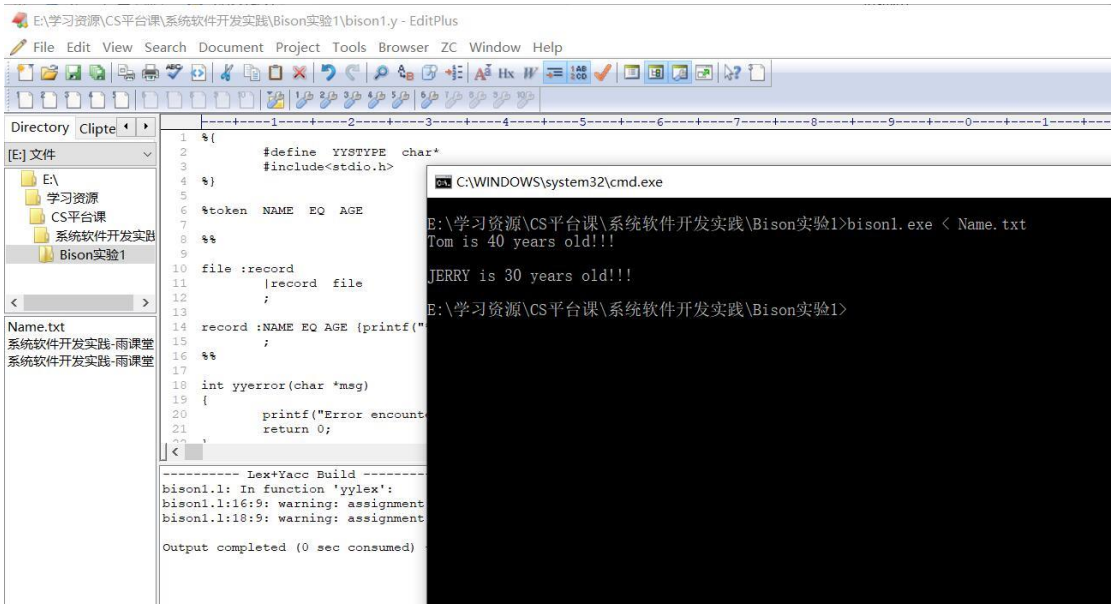

图 1-4 Win 下程序运行结果

1.5.2 CentOS 下运行结果

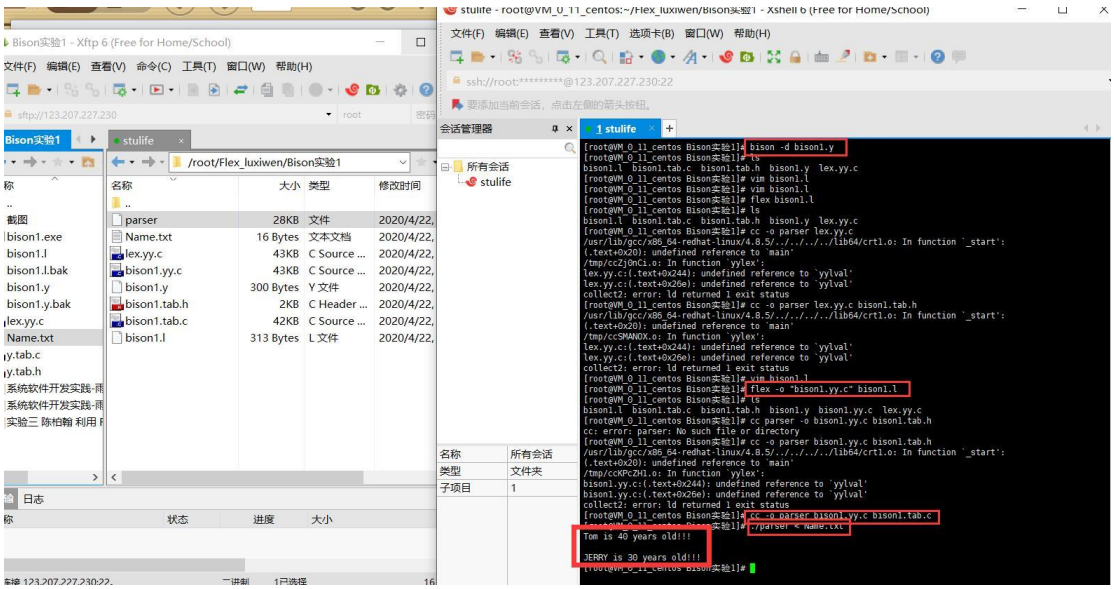

图 1-5 CentOS 下程序运行结果

1.6 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。

1.6.1 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 。

1.6.2 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 。

1.7 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。