**中国矿业大学计算机学院**

**2019 级本科生课程设计报告**

课程名称 系统软件开发实践

报告时间 2022.3.4

学生姓名 王杰永

学 号 03190886

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 | |  | |

Bison实验一

[1 实验内容 1](#_Toc97399249)

[2 Bison环境搭建 1](#_Toc97399250)

[2.1 Windows环境下Bison的安装与配置 1](#_Toc97399251)

[2.2 Linux环境下Flex的配置 2](#_Toc97399252)

[3 源码分析 2](#_Toc97399253)

[3.1 Name.y源码分析 3](#_Toc97399254)

[3.2 Name.l源码分析 4](#_Toc97399255)

[4 实验结果及分析 5](#_Toc97399256)

[5 实验总结 6](#_Toc97399257)

[5.1 你在编程过程中遇到了哪些难题？ 6](#_Toc97399258)

[5.2 你对你的程序的评价？ 7](#_Toc97399259)

[5.3 你的收获有哪些？ 7](#_Toc97399260)

* 1. 实验内容

1)阅读《Flex/Bison.pdf》第一章，第三章，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. Bison环境搭建
     1. Windows环境下Bison的安装与配置

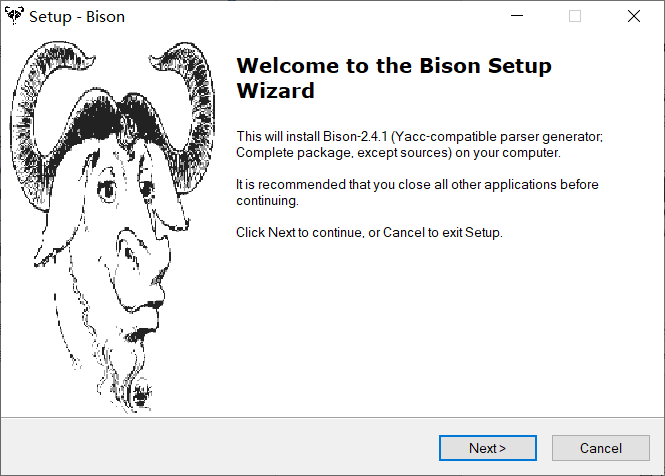
首先以管理员权限打开资料中的flex-2.5.4a-1.exe。

图1 Bison的安装

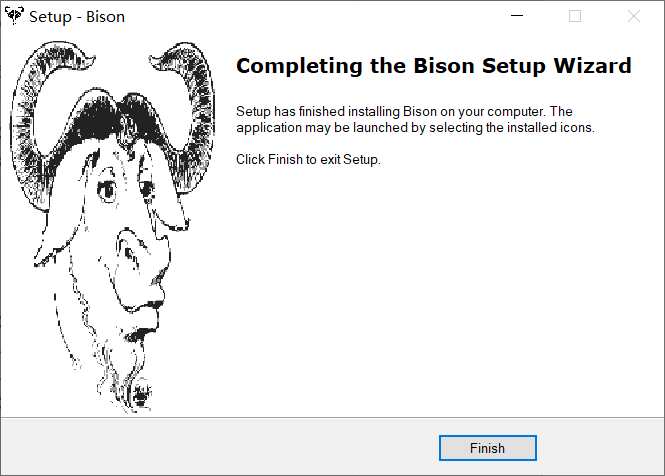
点击Next，接受许可协议，选择安装路径，即可安装成功。

图2 Bison安装成功

将安装好的flex根目录下的bin文件夹加入系统环境变量，以便在任何路径下都可以使用flex命令。

* + 1. Linux环境下Flex的配置

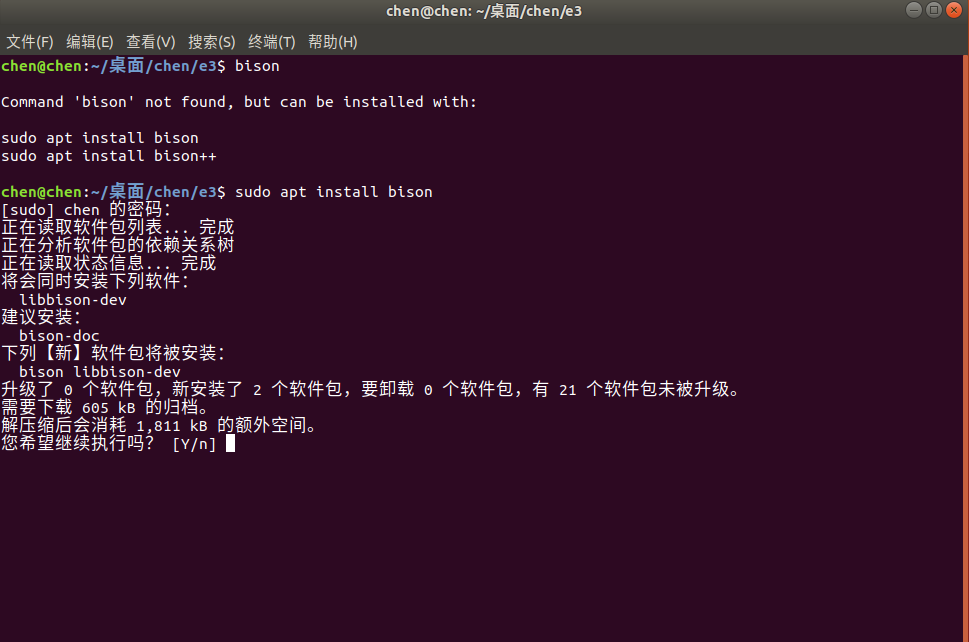
打开终端，使用命令sudo apt install bison，输入管理员密码，等待安装完成。

图3 ubuntu安装bison

图4 bison安装完成

* 1. 源码分析

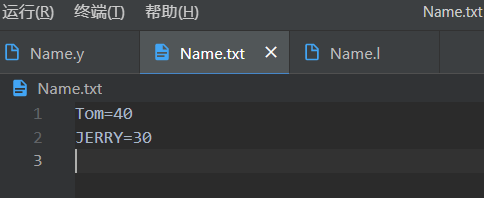
同Flex源码一样，一个Yacc程序也用双百分号分为了三段——声明、语法规则、C代码。本次实验以一个格式为 姓名 = 年龄 的文件作为例子，来说明语法规则。测试文件如下图所示。

图5 测试文件

* + 1. Name.y源码分析

资料中的语法分析源码文件Name.y内容如下。

1. %{
2. typedef char\* string;
3. #define YYSTYPE string
4. #include <stdio.h>
5. %}
7. %token NAME EQ AGE
9. %%
11. file: record
12. | record file
13. ;
14. record: NAME EQ AGE{printf("%s is %s years old!!!**\n**", $1, $3);}
15. ;
16. %%
18. int main()
19. {
20. yyparse();
21. return 0;
22. }
24. int yyerror(char\* msg){
25. printf("Error encountered: %s**\n**", msg);
26. return 0;
27. }

作为语法分析程序的生成程序，我们需要为语法规则定义一些符号（标记）。**行7**的语法便定义了NAME、EQ、AGE作为语法分析可以识别的token，也是文法的终结符。

当我们使用Flex进行词法分析时，每当输入串与Flex的一个正则表达式匹配时，对应的动作会返回当前匹配到的单词的token类型并赋值给Yacc中的yylval变量。由于yylval变量默认类型为int，本次实验我们需要的token类型为字符串，所以我们需要对yylval的类型YYSTYPE重定义。**行2、行3**将YYSTYPE重定义为字符串类型。

**行11至行14**定义了语法分析的语法规则。除了使用到已经定义的终结符外，在这里出现的file、record均是非终结符。

特别的，**行14**定义了产生式：

如果出现了NAME EQ AGE的可归约串，则会执行花括号中的动作。其中，$n表示产生式右侧的第n个符号的值。

* + 1. Name.l源码分析

1. %{
2. #include "Name.tab.h"
3. #include <string.h>
4. %}
6. char [A-Za-z]
7. num [0-9]
8. eq [=]
9. name {char}+
10. age {num}+
12. %%
13. {name} {
14. yylval = strdup(yytext);
15. return NAME;
16. }
17. {eq} {return EQ;}
18. {age} {
19. yylval = strdup(yytext);
20. return AGE;
21. }
22. %%
24. int yywrap(){return 1;}

将Lex与Yacc结合起来的办法，是在Flex的.l文件声明区引用Yacc文件生成的\*.tab.h文件。这样，在Flex的.l文件中可以使用定义的NAME、EQ、AGE等token，也可以使用yylval等特殊变量。

同时，在**行13至行21中**，每一个正则表达式的对应动作内，将匹配到的串的值赋给Yacc的yylval后，返回对应的token，以便于语法分析程序去处理。

* 1. 实验结果及分析

进入存放实验代码文件的目录，使用bison -d Name.y命令，生成Name.tab.c与Name.tab.h文件。

使用flex -oName.yy.c Name.l命令，生成Name.yy.c文件。

最后，使用gcc编译器对flex与bison文件联合编译。使用命令gcc Name.yy.c Name.tab.c -o Name.exe，生成可执行文件Name.exe。

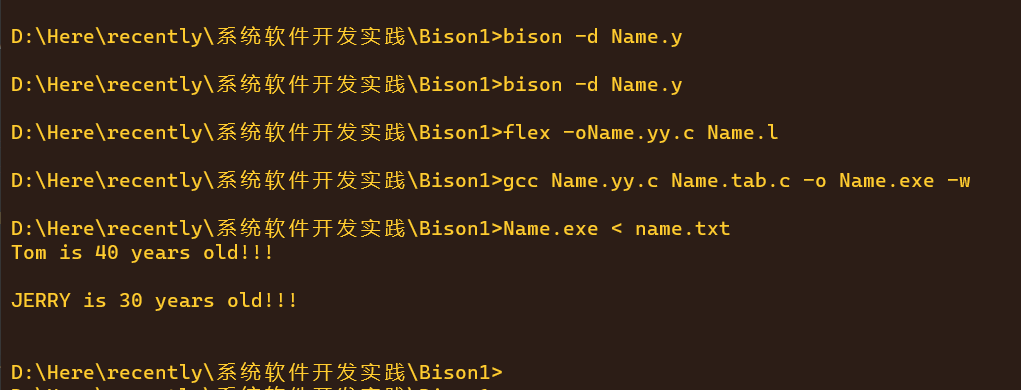
执行Name.exe < Name.txt，得到下图结果。

图6 windows实验结果

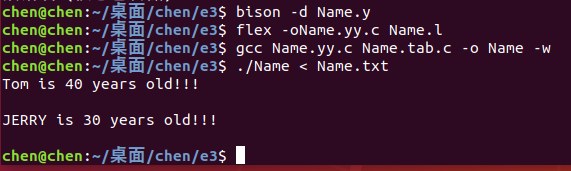
在Linux环境下，实验结果如下图所示。

图7 linux实验结果

* 1. 实验总结
     1. 你在编程过程中遇到了哪些难题？
        + bison命令执行时的报错

我在安装flex与bison时，均将其安装位置下的bin目录加入到了系统边境变量中，因此可以在任意路径下使用bison/flex命令。

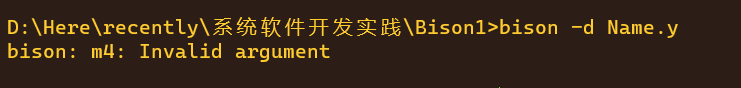
使用bison -d Name.y时，发生了如下错误：

图8 bison命令报错

通过查阅资料发现，bison安装位置的bin目录下存在可执行文件m4.exe，需要将该文件复制到程序执行路径下才可以继续执行bison命令。

* + - * 使用gcc联合编译Name.yy.c与Name.tab.c时各种warning的解决

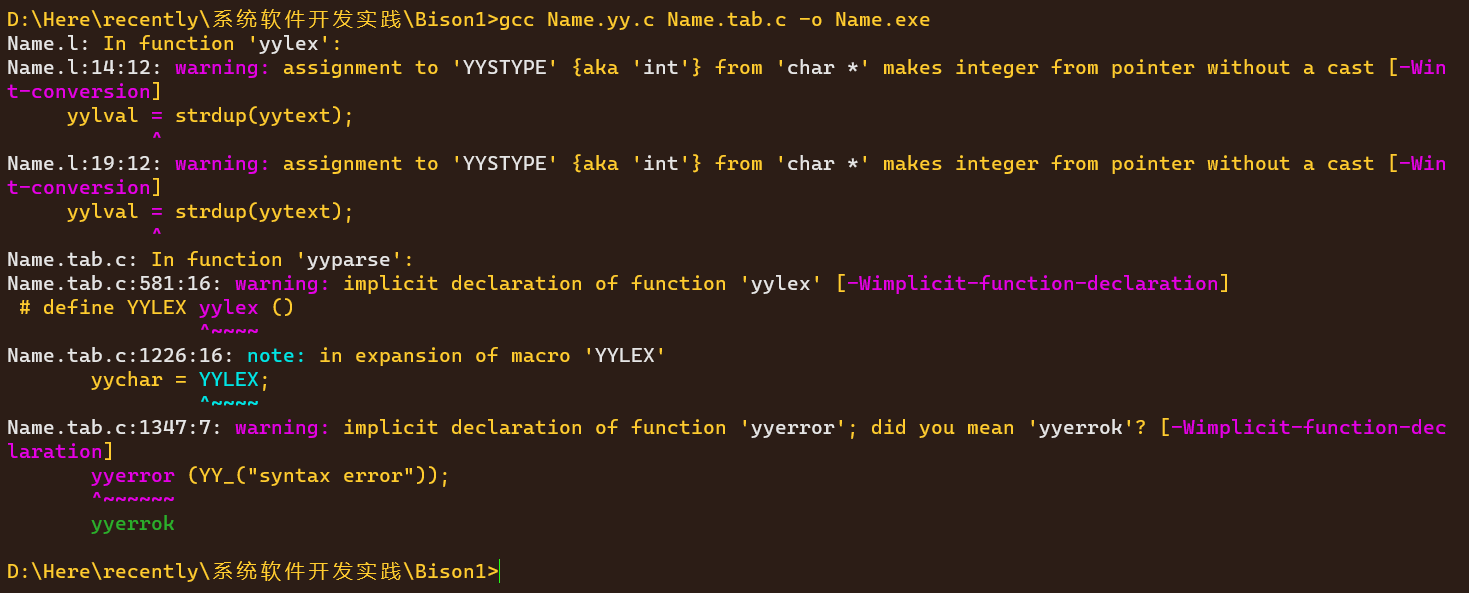
在windows环境下，使用gcc编译器，对lex与Yacc联合编译时，出现了大量警告信息。

图9 联合编译时的大量warning

可以看到，总共出现了4个warning，但是前两个warning的提示信息相同，因此一共有3个警告信息。

首先分析前两个警告信息：

|  |
| --- |
| Name.l:14:12: warning: assignment to 'YYSTYPE' {aka 'int'} from 'char \*' makes integer from pointer without a cast [-Wint-conversion]  yylval = strdup(yytext);  ^  Name.l:19:12: warning: assignment to 'YYSTYPE' {aka 'int'} from 'char \*' makes integer from pointer without a cast [-Wint-conversion]  yylval = strdup(yytext);  ^ |

对于这两个警告信息，大体意思是对于类型YYSTYPE，使用char\*覆盖了int类型而没有进行任何转换。通过查询相关资料了解到，YYSTYPE默认为int类型，我们在Yacc文件中将其覆盖为char\*类型，从而导致了这个warning。但是，本次实验的token理应使用char\*类型表示，因此可以将生成的Name.tab.c文件中的typedef int YYSTYPE改为typedef char\* YYSTYPE。从而解决警告信息。

接下来分析后两个警告信息。

Name.tab.c: In function 'yyparse':

Name.tab.c:581:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]

# define YYLEX yylex ()

^~~~~ ^

Name.l:19:12: warning: assignment to 'YYSTYPE' {aka 'int'} from 'char \*' makes integer from pointer without a cast [-Wint-conversion]

yylval = strdup(yytext);

^

这属于c语言中多文件编程中，某些文件缺少了函数声明导致的。

在Name.y文件中，声明区加入int yylex()与intyyerror()的声明即可。

* + 1. 你对你的程序的评价？

本次实验完成的语法分析器规则较为简单，和之前的实验相比加入了语法分析的环节。在基于bison的语法分析程序的编写上，还需要更多努力。

* + 1. 你的收获有哪些？

通过本次实验，我学会了bison的.y文件的编写规则，用bison做语法分析，其代码的简洁以及良好的可拓展性让我对这个工具产生了浓厚的兴趣。相比于上学期完全基于c语言的词法/简单语法分析器来说，本次仅用少量代码就可以完成更为复杂的功能。受益匪浅。