中国矿业大学

CHINA UNIVERSITY OF MINING AND TECHNOLOGY

中国矿业大学计算机学院

2017级本科生课程设计报告

实验三 Bison实验1

课程名称　系统软件开发实践

报告时间　2020.04.20

学　　号

目录

实验三 Bison实验1

1 实验目的

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。

2 分析代码

2.1 分析Flex源代码

```
1   %option noyywrap
2   %{
3       #include "Name.tab.h"
4       #include <stdio.h>
5       char *strdup(const char *s);
6   %}
7
8   char [A-Za-z]
9   num [0-9]
10  eq [=]
11  name {char}+
12  age {num}+
13
14  %%
15  {name} {
16      yylval.sval = strdup(yytext);
17      return NAME;
18  }
19  {eq} { printf("eq\t%s\n",yytext);return EQ; }
20  {age} {
21      yylval.sval = strdup(yytext);
22      return AGE;
23  }
24  %%
```

代码分为两部分，第一部分1-6行，包含声明和选项，第1行 `%option noyywrap` 要求它不使用 yywrap；3-4行中的代码会被照抄在生成的C文件开头，这里面加载了自定义 `Name.tab.h` 头文件和系统 `stdio.h` 头文件，并且预定义了一个 `strdup` 函数，用于指定参数与返回类型。

第二部分8-23行，8-12行，主要是一些匹配模式，用 char 匹配字母，num 匹配数字，eq 匹配等于号，name 匹配多个字母，age 匹配多个数字。15-23行是匹配成功时需要执行的C代码，匹配 name 会将匹配文本保存在 yylval.sval，并返回此匹配值到 NAME，eq 与 age 同理。

2.2 分析Bison源代码

```
1  %{
2      #include<stdio.h>
3      #include <stdlib.h>
4      extern int yylex();
5      extern int yyparse();
6      int yyerror(char* msg);
7  %}
8
9  %union {
10     char *sval;
11 }
12 %token<sval> NAME AGE
13 %token EQ
14
15 %start file
16
17 %%
18 file: record
19 | record file
20 ;
21 record: NAME EQ AGE {printf("%s is %s years old!!\n", $1, $3);}
22 ;
23 %%
24
25 int main(){
26     yyparse();
27     return 0;
28 }
29 int yyerror(char *msg){
30     printf("Error encountered: %s \n", msg);
31     return 0;
32 }
```

bison程序包含了与 flex 程序相同的三部分结构：声明部分、规则部分和C代码部分。

第一部分1-13行，包含了会被原样拷贝到目标程序开头的C代码，同样也通

过 %{ 与 }% 来声明。这里加载了头文件 stdio.h 与 stdlib.h，预定义了外部函数 yylex 与 yyparse，同样预定义了一个函数 yyerror 指明参数类型和数量与返回值。

9-13行是记号声明，以便于告诉 bison 在语法分析程序中记号的名称。通常记号总是使用大写，任何没有声明为语法符号必须出现在至少一条规则的左边。

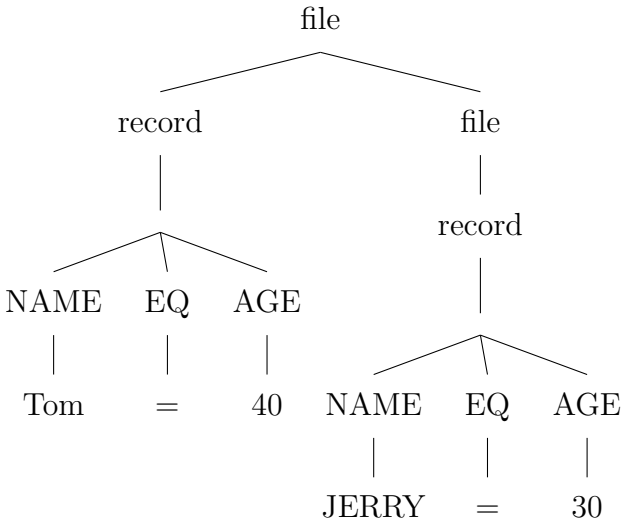
%union 声明指定语义值的可能数据类型的整个集合。关键字 %union 后跟带有括号的代码，这些代码包含与C中的联合内部相同的内容。%union 声明指定语义值的可能数据类型的整个集合。关键字 %union 后跟带有括号的代码，这些代码包含与C中的联合内部相同的内容。

缺省情况下，bison 假定语法的开始符号是语法规范部分中指定的第一个非终止符，这里使用 %start 声明覆盖此限制。

第二部分9-13行，包含了通过简单 BNF 定义的规则。bison 使用单一的冒号，分号用作表示规则的结束。同样，C的动作代码在每条规则之后用花括号括起。每个 bison 规则中的语法符号都有一个语义值，目标符号（冒号左边的语法符号）的值在动作中用 $$ 代替，右边的语法符号的语义值依次为 $1、$2，直到这条规则的结束。当词法分析器返回记号时，记号值总是存储在 yyval 里，其他语法符号的语义值则在语法分析器的规则里进行设置。这里面没有使用 $$ ，仅仅输出 $1、$2 等的值。

第三部分25-32行，包含了用户自定义函数与语句，主函数调用 yyparse()，开始分析程序，分析成功，返回0，否则调用 yyerror()，输出错误原因。

语法分析树如下所示：



3 在Windows环境下实验

3.1 安装Bison

Bison安装页面如图1所示，安装结果如图2所示。

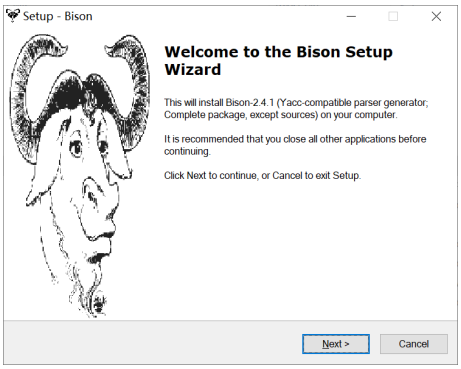


图 1: Bison安装界面

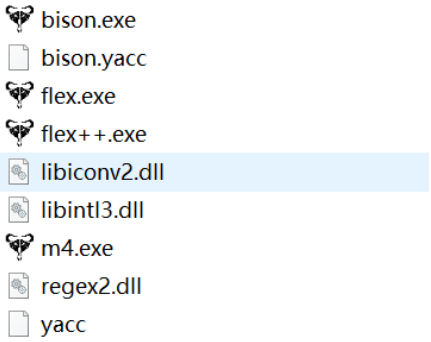


图 2: Bison安装结果

3.2 实验过程

将代码 Name.l、Name.y 与 Name.txt 放在实验目录 ex3 下，打开 Developer Command Prompt for VS 2019，由于实验1时已经将此实验目录加入环境变量，这里直接输入 `bison -d Name.y`，生成文件 `Name.tab.c` 和 `Name.tab.h` ，结果如图3所示。

然后在命令行再输入 `flex Name.l`，生成文件 `lex.yy.c`，结果如图4 所示。

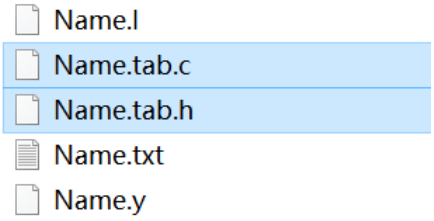


图 3: bison运行结果

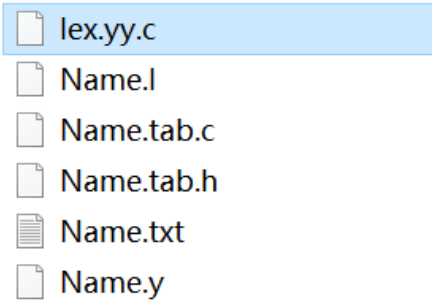


图 4: flex运行结果

之后，进行c语言代码编译，明两行输入 `cl lex.yy.c Name.tab.c`，结果如图5所示，生成了可执行文件 `lex.yy.exe`。

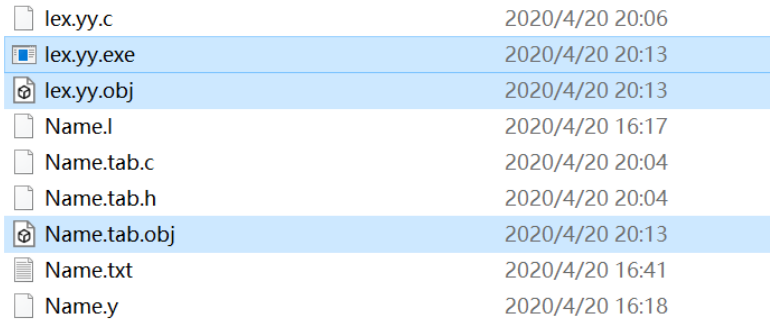


图 5: 编译结果

整个过程命令行结果如图6所示，编译成功。

```
C:\Users\yuan\Documents\课程\系统软件开发实践\ex3>bison -d Name.y

C:\Users\yuan\Documents\课程\系统软件开发实践\ex3>flex Name.l

C:\Users\yuan\Documents\课程\系统软件开发实践\ex3>cl lex.yy.c Name.tab.c
用于 x86 的 Microsoft (R) C/C++ 优化编译器 19.24.28316 版
版权所有(C) Microsoft Corporation。保留所有权利。

lex.yy.c
Name.tab.c
正在生成代码...
Microsoft (R) Incremental Linker Version 14.24.28316.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:lex.yy.exe
lex.yy.obj
Name.tab.obj
```

图 6: 命令行结果

3.3 实验结果

命令行输入 `lex.yy.exe < Name.txt`，结果如图7所示，与分析结果一致，输出 `Tom is 40 years old!!!` 与 `JERRY is 30 years old!!!`。

```
C:\Users\yuan\Documents\课程\系统软件开发实践\ex3>lex.yy.exe < Name.txt
Tom is 40 years old!!!

JERRY is 30 years old!!!

C:\Users\yuan\Documents\课程\系统软件开发实践\ex3>
```

图 7: windows实验结果

4 在ubuntu环境下实验

4.1 安装Bison

在终端输入 `sudo apt install bison` 然后再输入密码，就可以完成 bison 的安装，安装过程与结果如图8所示。

4.2 实验过程

将代码 Name.l、Name.y 与 Name.txt 放在实验目录 ex3 下，打开终端，由于 ubuntu 自动将 bison 加入环境变量，这里直接输入 `bison -d Name.y`，生成文件 `Name.tab.c` 和 `Name.tab.h` ，结果如图9所示。

然后在命令行再输入 `flex Name.l`，生成文件 `lex.yy.c`，结果如图10所示。

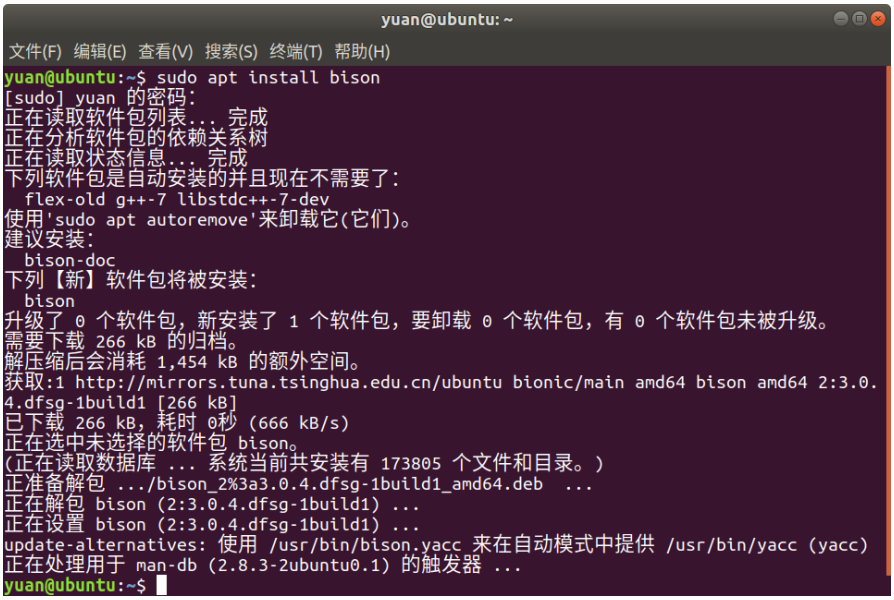


图 8: 安装Bison

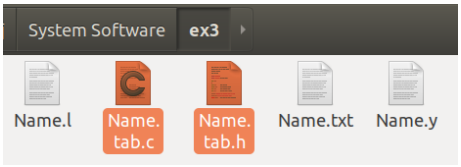


图 9: bison运行结果

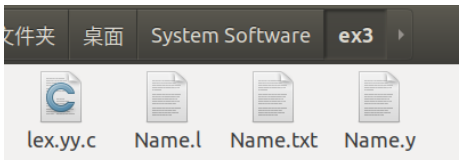


图 10: flex运行结果

之后，进行c语言代码编译，明两行输入 `cc lex.yy.c Name.tab.c`，结果如图11所示，生成了文件 `a.out`。

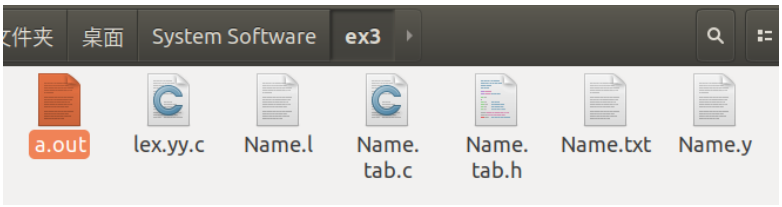
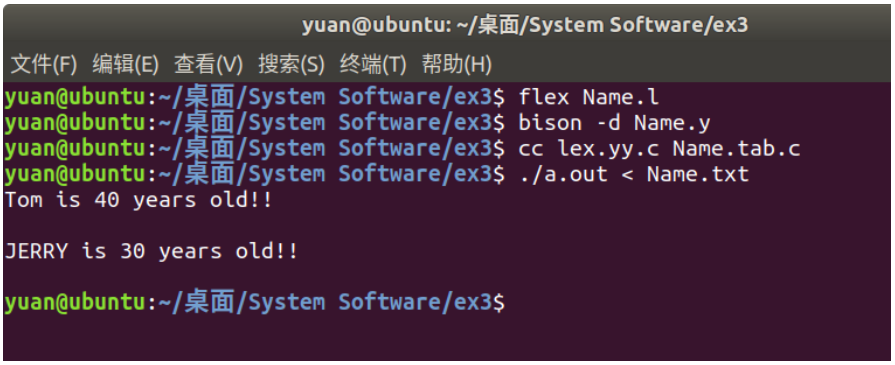


图 11: 编译结果

4.3 实验结果

命令行输入 `./a.out < Name.txt`，结果如图12所示，与分析结果一致，输出 `Tom is 40 years old!!!` 与 `JERRY is 30 years old!!!`。

5 实验感想


yuan@ubuntu: ~/桌面/System Software/ex3
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
yuan@ubuntu:~/桌面/System Software/ex3$ flex Name.l
yuan@ubuntu:~/桌面/System Software/ex3$ bison -d Name.y
yuan@ubuntu:~/桌面/System Software/ex3$ cc lex.yy.c Name.tab.c
yuan@ubuntu:~/桌面/System Software/ex3$ ./a.out < Name.txt
Tom is 40 years old!!
JERRY is 30 years old!!
yuan@ubuntu:~/桌面/System Software/ex3$