

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
2 Bison 环境搭建	1
2.1 Windows 环境下 Bison 的安装与配置.....	1
2.2 Linux 环境下 Flex 的配置.....	2
3 源码分析.....	2
3.1 Name.y 源码分析	3
3.2 Name.l 源码分析	4
4 实验结果及分析.....	5
5 实验总结.....	6
5.1 你在编程过程中遇到了哪些难题?	6
5.2 你对你的程序的评价?	7
5.3 你的收获有哪些?	7

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。

2 Bison 环境搭建

2.1 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。

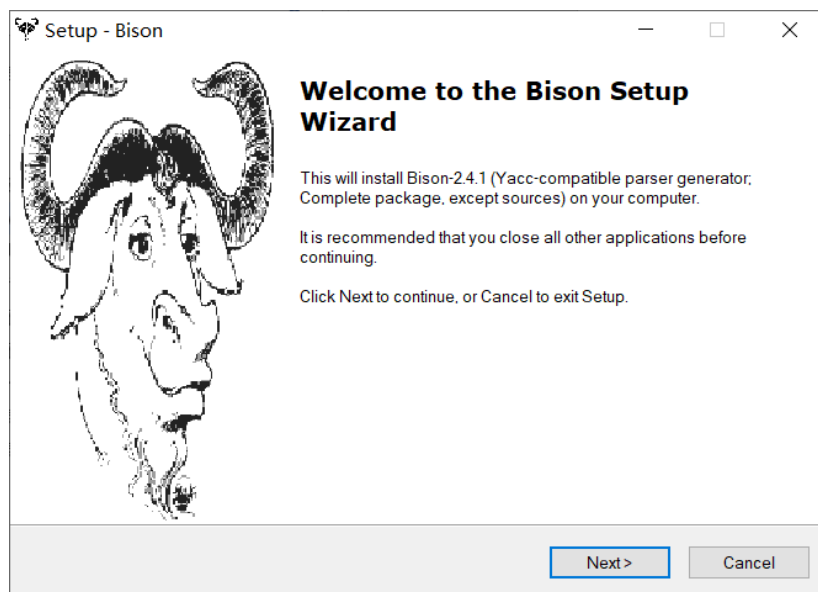


图 1 Bison 的安装

点击 Next，接受许可协议，选择安装路径，即可安装成功。

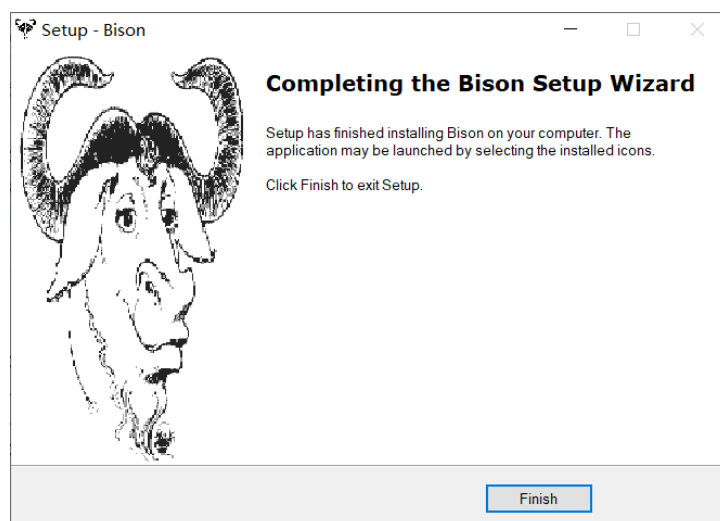
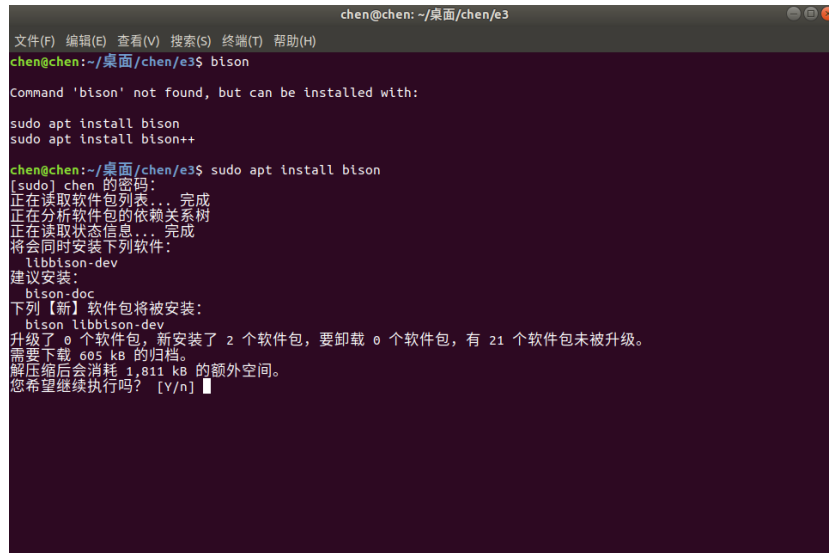


图 2 Bison 安装成功

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。

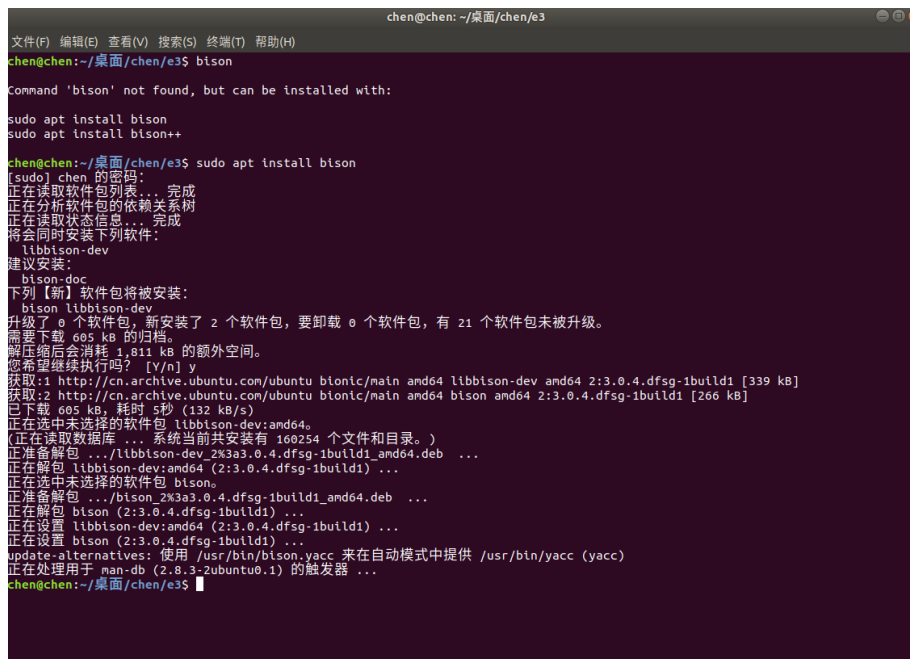
2.2 Linux 环境下 Flex 的配置

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。



```
chen@chen: ~/桌面/chen/e3
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
chen@chen:~/桌面/chen/e3$ bison
Command 'bison' not found, but can be installed with:
sudo apt install bison
sudo apt install bison++
chen@chen:~/桌面/chen/e3$ sudo apt install bison
[sudo] chen 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件:
libbison-dev
建议安装:
bison-doc
下列【新】软件包将被安装:
bison libbison-dev
升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 21 个软件包未被升级。
需要下载 605 kB 的归档。
解压后会消耗 1,811 kB 的额外空间。
您希望继续执行吗？ [Y/n]
```

图 3 ubuntu 安装 bison



```
chen@chen: ~/桌面/chen/e3
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
chen@chen:~/桌面/chen/e3$ bison
Command 'bison' not found, but can be installed with:
sudo apt install bison
sudo apt install bison++
chen@chen:~/桌面/chen/e3$ sudo apt install bison
[sudo] chen 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件:
libbison-dev
建议安装:
bison-doc
下列【新】软件包将被安装:
bison libbison-dev
升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 21 个软件包未被升级。
需要下载 605 kB 的归档。
解压后会消耗 1,811 kB 的额外空间。
您希望继续执行吗？ [Y/n] y
获取:1 http://cn.archive.ubuntu.com/ubuntu bionic/main amd64 libbison-dev amd64 2:3.0.4.dfsg-1build1 [339 kB]
获取:2 http://cn.archive.ubuntu.com/ubuntu bionic/main amd64 bison amd64 2:3.0.4.dfsg-1build1 [266 kB]
已下载 605 kB，耗时 5 秒 (132 kB/s)
正在选中未选择的软件包 libbison-dev:amd64。
(正在读取数据库 ... 系统当前共安装有 160254 个文件和目录。)
正在准备解包 .../libbison-dev_2k3a3.0.4.dfsg-1build1_amd64.deb ...
正在解包 libbison-dev:amd64 (2:3.0.4.dfsg-1build1) ...
正在选中未选择的软件包 bison。
正在解包 bison_2k3a3.0.4.dfsg-1build1_amd64.deb ...
正在解包 bison (2:3.0.4.dfsg-1build1) ...
正在设置 libbison-dev:amd64 (2:3.0.4.dfsg-1build1) ...
正在设置 bison (2:3.0.4.dfsg-1build1) ...
update-alternatives: 使用 /usr/bin/bison.yacc 来在自动模式中提供 /usr/bin/yacc (yacc)
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...
chen@chen:~/桌面/chen/e3$
```

图 4 bison 安装完成

3 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。

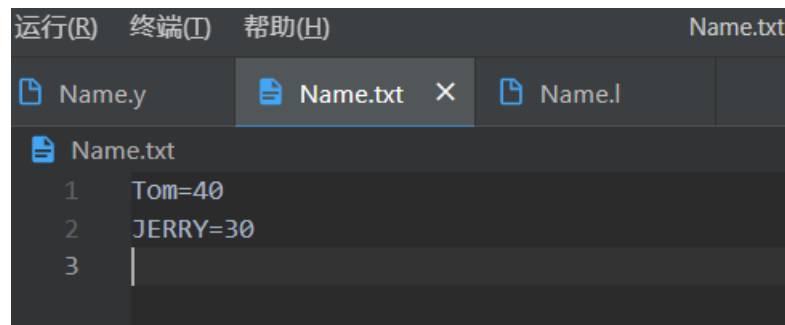


图 5 测试文件

3.1 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. %}
6.
7. %token NAME EQ AGE
8.
9. %%
10.
11. file: record
12.     | record file
13.     ;
14. record: NAME EQ AGE{printf("%s is %s years old!!!\n", $1, $3);}
15.     ;
16. %%
17.
18. int main()
19. {
20.     yyparse();
21.     return 0;
22. }
23.
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：

record \rightarrow *NAME EQ AGE* {printf("%s is %s years old!!!\n", \$1, \$3);}

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。

3.2 Name.l 源码分析

```
1. %{
2.     #include "Name.tab.h"
3.     #include <string.h>
4. }%
5.
6. char    [A-Za-z]
7. num     [0-9]
8. eq      [=]
9. name    {char}+
10. age     {num}+
11.
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. %%
```

```
23.  
24. int yywrap(){return 1;}  
25.
```

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。

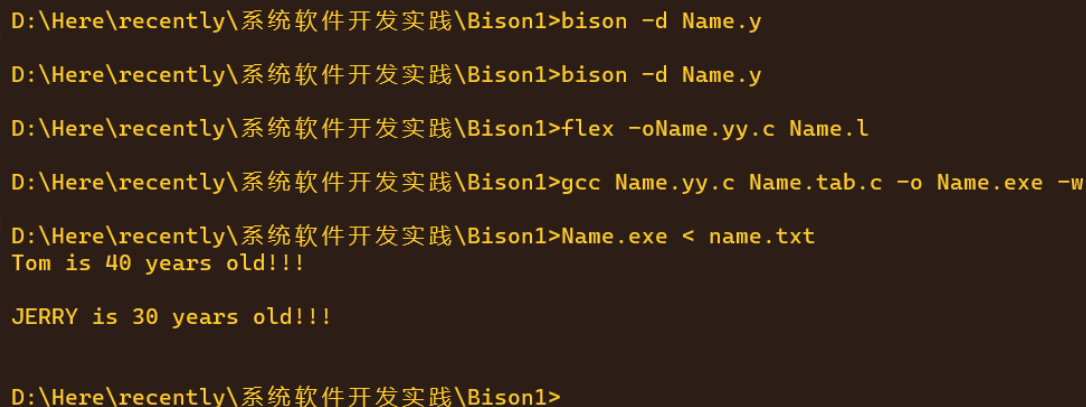
4 实验结果及分析

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 -w`，生成可执行文件 Name.exe。

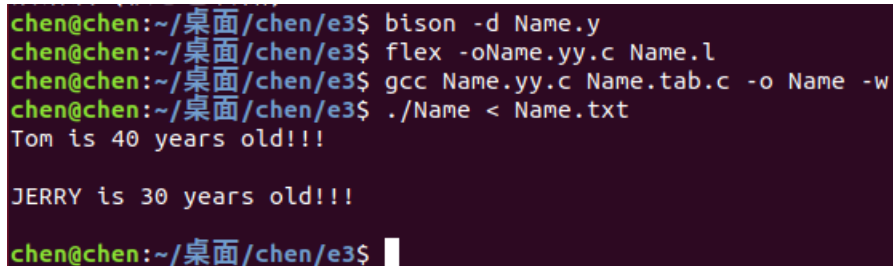
执行 `Name.exe < Name.txt`，得到下图结果。



```
D:\Here\recently\系统软件开发实践\Bison1>bison -d Name.y  
D:\Here\recently\系统软件开发实践\Bison1>bison -d Name.y  
D:\Here\recently\系统软件开发实践\Bison1>flex -oName.yy.c Name.l  
D:\Here\recently\系统软件开发实践\Bison1>gcc Name.yy.c Name.tab.c -o Name.exe -w  
D:\Here\recently\系统软件开发实践\Bison1>Name.exe < name.txt  
Tom is 40 years old!!!  
  
JERRY is 30 years old!!!  
  
D:\Here\recently\系统软件开发实践\Bison1>
```

图 6 windows 实验结果

在 Linux 环境下，实验结果如下图所示。



```
chen@chen:~/桌面/chen/e3$ bison -d Name.y  
chen@chen:~/桌面/chen/e3$ flex -oName.yy.c Name.l  
chen@chen:~/桌面/chen/e3$ gcc Name.yy.c Name.tab.c -o Name -w  
chen@chen:~/桌面/chen/e3$ ./Name < Name.txt  
Tom is 40 years old!!!  
  
JERRY is 30 years old!!!  
  
chen@chen:~/桌面/chen/e3$
```

图 7 linux 实验结果

5 实验总结

5.1 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 目录加入到了系统环境变量中,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:

```
D:\Here\recently\系统软件开发实践\Bison1>bison -d Name.y
bison: m4: Invalid argument
```

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。

```
D:\Here\recently\系统软件开发实践\Bison1>gcc Name.yy.c Name.tab.c -o Name.exe
Name.l: In function 'yylex':
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);
           ^
Name.tab.c: In function 'yyparse':
Name.tab.c:581:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
    # define YYLEX yylex ()
                   ^~~~~~
Name.tab.c:1226:16: note: in expansion of macro 'YYLEX'
    yychar = YYLEX;
             ^~~~~~
Name.tab.c:1347:7: warning: implicit declaration of function 'yyerror'; did you mean 'yyerrok'? [-Wimplicit-function-declaration]
    yyerror (YY_("syntax error"));
    ^~~~~~
    yyerrok
D:\Here\recently\系统软件开发实践\Bison1>
```

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]
```

```

    yynlval = strdup(yytext);
    ^
Name.1:19:12: warning: assignment to 'YYSTYPE' {aka 'int'} from 'char
*' makes integer from pointer without a cast [-Wint-conversion]

    yynlval = 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.1:19:12: warning: assignment to 'YYSTYPE' {aka 'int'} from 'char *'
makes integer from pointer without a cast [-Wint-conversion]

    yynlval = 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 int yyerror() 的声明即可。

5.2 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。

5.3 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。