用 GO 语言实现 C 语言词法分析器

施海彬
（厦门大学信息科学与技术学院　福建　厦门　361005）

【摘　要】:词法分析器是编译器的第一阶段,其从左到右读入程序的源文件,而后分解出有意义的词法单元。大多数编译器的词法分析器都采用手工实现,所以学习、实现词法分析器在教学上有重要意义。本系统使用新的系统级程序语言 GO 语言,并利用其内置的字典数据类型,构建出几个查找表,简化了程序,实现了 C 语言的词法分析器。

【关键字】:词法分析;编译程序;GO 语言

1. 序言

编译器在计算机科学中有着重要的作用。词法分析器是编译程序的第一阶段,其通过读入源程序的字符流,把这些一个个连续的字符分解成有意义的词法单元,而后输出给语法分析器。又由于词法分析器相对简单性,大多数编译器实现都采用手工实现,而不是自动实现,所以理解并实现词法分析器,是一个重要的技能。而 GO 语言是一个新的极具潜力的系统级语言,两者结合,即用 GO 语言实现 C 语言词法分析器,在教学上有重要意义,可帮助学生学习 GO 语言的同时,帮助掌握和设计实现词法分析器。

2. GO 语言介绍

GO 语言是谷歌 2009 年发布的系统级程序语言。谷歌之所开发该语言,因自重要的系统级编程语言诞生距今已超过十几年了,这段时间计算机世界已经发生了很大的变化,尤其过去十多年间软件开发的难度令人沮丧,仍没有哪种主流的编程语言能很好适合软件的大型并行开发。例 C 语言是简单,但不适合大型开发;C++语言过于复杂;而 java 语言性能有问题,繁琐,并也逐渐走向复杂;python 灵活,但性能也有问题。

GO 语言简洁明了,需要记的语言细节少,能快速编译、快速执行,可用于编写核心层和驱动层,而以往这些都是 C 或 C++的支配领域;GO 语言灵活,内置类 Python 的 list,map 等常用数据结构,使得开发快速;GO 语言大量简化了指针,更安全,绝大部分问题都能消灭在编译过程中;完善的模块支持,友好的并行架构,适合于大型软件的并行开发,尤其适合云计算的今天。GO 语言可以说结合了 C 语言和 python 语言的优点。

3. C 语言词法单元

词法分析器的功能就是从左到右读入程序语言的源文件,而后以尽量最长为原则形成词法单元流。C 语言的词法单元由以下 5 类组成:标识符、关键字、常量、运算符和分隔符,另外两个特殊结构:注释、预处理器。

⑴ 标识符由大小写拉丁字母、数字和下划线组成的序列,但不能以数字开头。与关键字相同的标志符则作为关键字,如 if、int、then 等等。

⑵ 常量分为四种不同的类型:整型(如 14、0x13 等)、浮点型(如 3.14、1.0e3 等)、字符型(如'A'、'B'等)和字符串型(如"copy"、"enter"等)。

⑶ 运算符,如+、>、<等。

⑷ 分隔符为(、)、{、}、…、,、;、:。

⑸ 注释作为空格处理,其分为多行注释和单行注释。多行注释由/* 开头,以 */结束。单行注释由//开始,至行结束。

⑹ 预处理器用特殊的预处理命令行控制,这些命令以 # 打头。

4. 设计实现

4.1 查找表

本系统构建了几个查找表,用于简化程序,尤其 GO 语言内置实现了字典数据类型,使得查找表的构建方式更为简单。

字符类型查找表,由一个数组构成的查找表实现,用于把字符的数值映射为 8 种类型:运算符、数字、字母、空格、回车、单引号、双引号、null(非法字符)。如把'A'映射为字母类型,'0'映射为数字类型。

```
var clang_char= [..] byte {
        char_sort_null,
```

```
char_sort_null,
char_sort_null,
char_sort_null,
...}
```

关键字查找表(包括预处理命令),使用字典方式实现。标识符在该字典中查找到的就是关键字,以与一般标识符区分开。

```
var clang_key_map=map [string] int{
    "inline":clang_key_inline,
    "asm":clang_key_asm,
    "char":clang_key_char,
    "continue":clang_key_continue,
    "#endif":clang_key_pre_endif,
    "static":clang_key_static,
    ...}
```

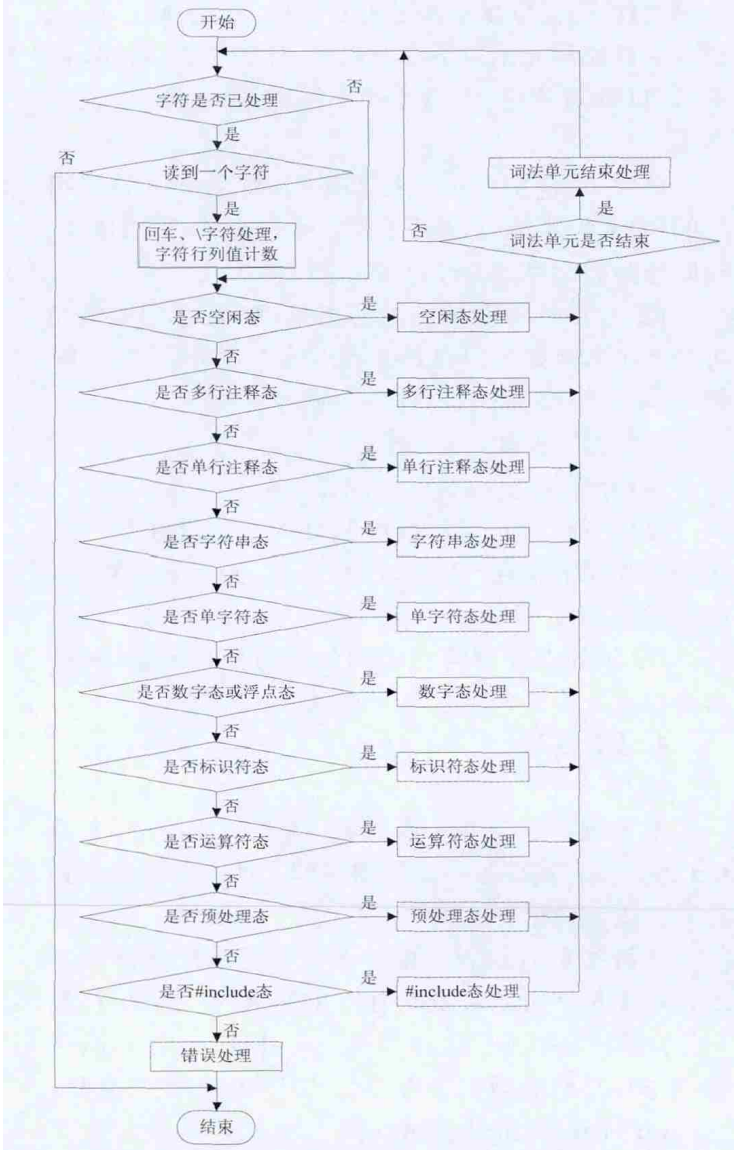

图 1 主流程图

运算符查找表(包括分隔符,但不包含单引号和双引号),使用字典方式实现。若运算符在该字典中查找到,则说明存在;若获得的第 1 项为"1",说明该运算符识别结束;若获得的第 1 项为"2",则运算符识别尚未结束,还需要读入新字符来判断。如刚读入"+",则可能是"+",或者是"++",或者是"+=",所以还需要再读入新字符才能判断。

```
var clang_op_map=map [string] cClangOpNums {
    ":":{2,clang_key_op_colon},
    ";":{1,clang_key_op_semi},
    "<":{2,clang_key_op_less},
    "=":{2,clang_key_op_assign},
    ">>=":{1,clang_key_op_right_move_assign},
    "<<":{2,clang_key_op_left_move},
    ...}
```

4.2 主流程

本系统的主流程图见图 1,其首先读取一个字符,而后跳到当前的词法单元状态去处理该字符;处理完该字符后就跳出到读取字符处。本系统的处理方式并非是传统方式,在某个词法单元识别循环中不断读取字符,至该词法单元识别完成,再跳出到主程序。主要是想给学生留下些修改空间,学生能自己比较两种方式的优劣。

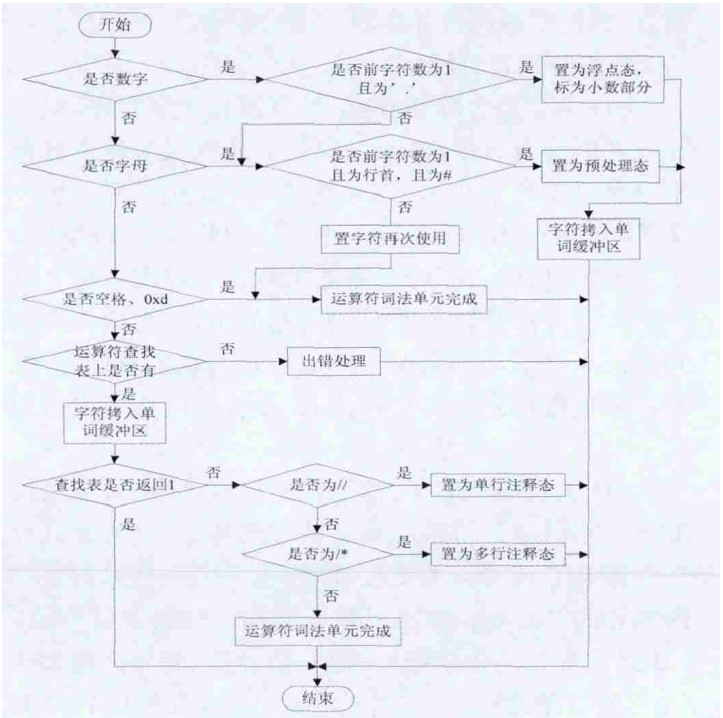

图 2 运算态流程图

4.3 运算符态处理

运算态处理函数的流程图见图 2,若输入是运算

符（或者分隔符），会查找 clang_op_map 表，而后看其返回值在进行相应处理。

4.4 数字态处理

数字态处理函数是词法单元识别中最复杂的，其要处理 8 进制、16 进制、浮点数，以及长整数的结尾 u、l 等等，其流程图见图 3。

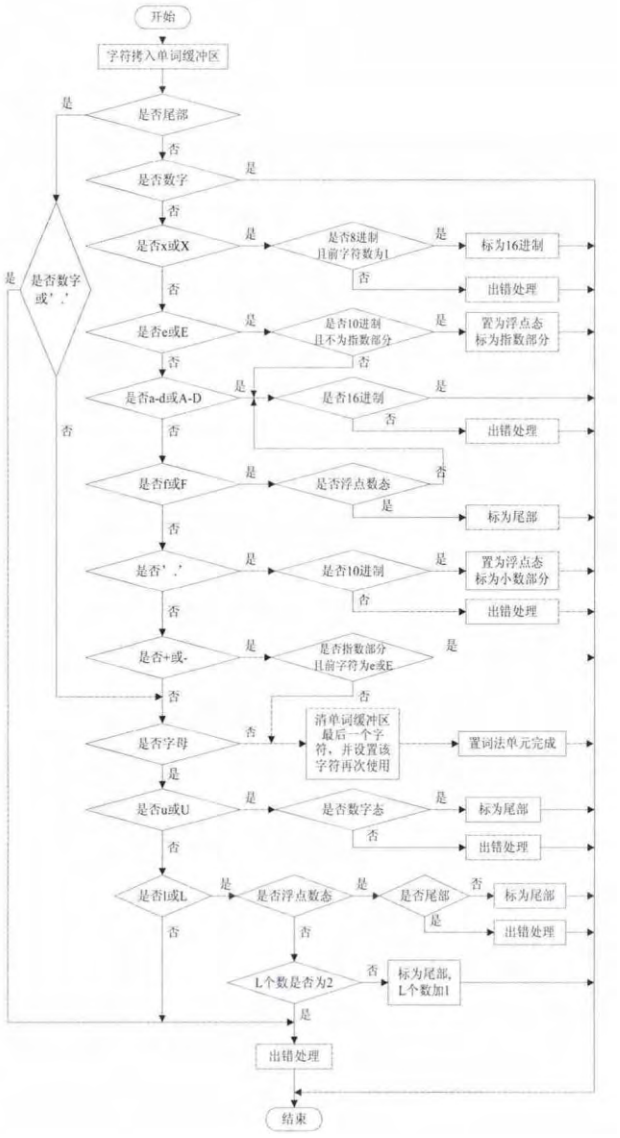

图 3 数字态流程图

4.5 其他处理过程

空闲态处理函数，根据读入的类型为运算符、数字、字母、空格、回车、单引号、双引号等不同类型，把状态设置为相应的状态。数字若为 0，则还要标为 8 进制；而回车类、空格类型则忽略。

标识符态处理函数，对输入字符进行判断，若为字母或者数字则继续退出去读下一个字符；否则识别完成，并查找关键字 clang_key_map 字典，若有存在，则该字符串为关键字。

预处理态处理函数，对输入字符进行判断，若不为字母，则识别完成，并且也查找关键字 clang_key_map 字典，若不存在，则错误处理；若为“#include”，则进入 #include 态。新增 #include 态，主要是想把“<文件名>”作为一个词法单元。

字符串态处理函数，对输入字符进行双引号判断，若是双引号，且“\”不为有效，则识别完成。单字符态、多行注释态、单行注释态处理函数也有相应的处理过程。

4.6 结果输出

对本系统以大量 linux 内核源代码文件进行了测试，均未出错，结果也验证正确。如用 linux 内核源代码“include/linux/cpumask.h”文件做测试，该文件第 78 行代码为“extern const struct cpumask *const cpu_possible_mask;”，输出如下：

```
...
78,1,0,extern
78,9,0,const
78,15,0,struct
78,22,4,cpumask
78,30,6,*
78,31,0,const
78,37,4,cpu_possible_mask
78,53,6,;
...
```

其第 1 列为行号；第 2 列为列号；第 3 列为该词法单元的类型：关键字为 0，整数为 1，浮点数为 2，字符串为 3，标识符为 4，字符为 5，运算符为 6，预处理命令为 7。

5. 结语

本文用 GO 语言设计并实现了 C 语言的词法分析器，利用 GO 语言内置的字典数据类型实现了查找表，使得程序更为简单。同时，本系统教学上可让学生参考、做实验，以及做二次开发。

参考文献：

[1]孙悦红. 编译原理及实现[M].清华大学出版社, 2005
[2](美)Harbison. C 语言参考手册(第五版)[M].机械工业出版社, 2003
[3]温敬和. 编译原理实用教程[M].清华大学出版社,2005
[4]http://golang.org/