NCTU CS 0516076 施威綸

Compiler Project 4

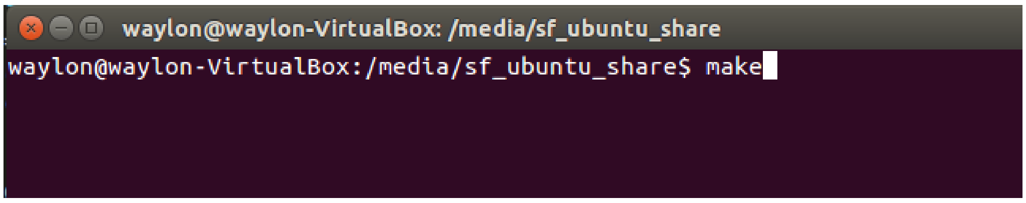
Project: C- compiler semantic analyzer — type checking

Date: December 26, 2018

Platform: Linux Ubuntu14.04

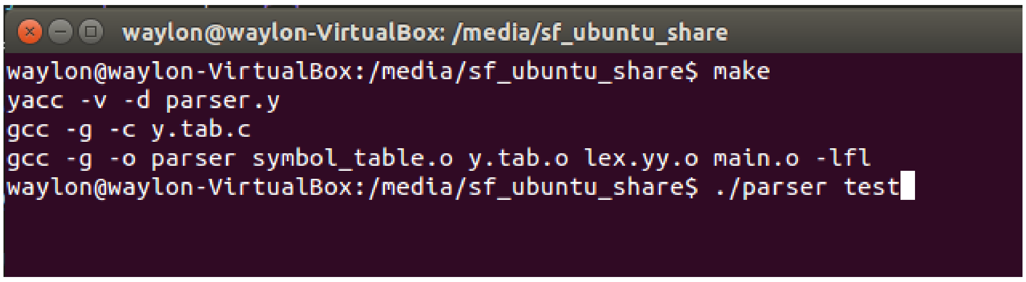
How to run:

1. Use make to compile the scanner and the parser



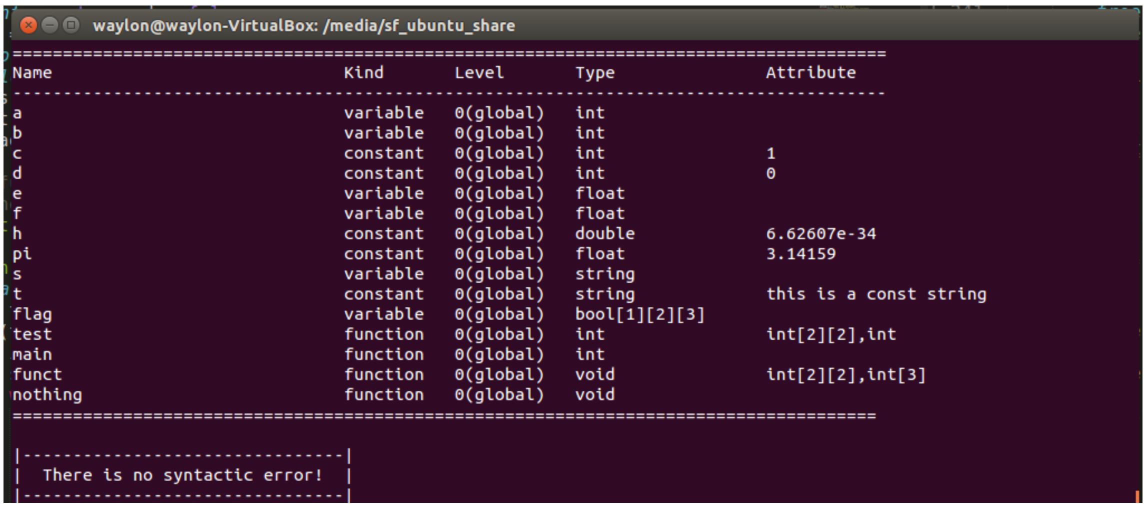
```
waylon@waylon-VirtualBox:/media/sf_ubuntu_share$ make
```

2. Execute the parser with testfile(test)



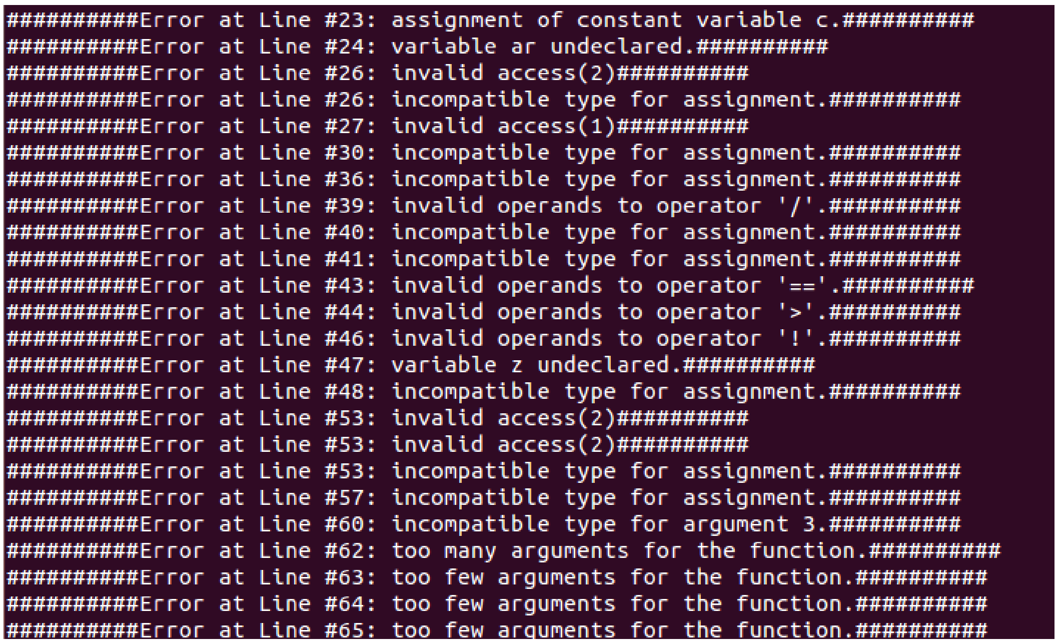
```
waylon@waylon-VirtualBox:/media/sf_ubuntu_share$ make
yacc -v -d parser.y
gcc -g -c y.tab.c
gcc -g -o parser symbol_table.o y.tab.o lex.yy.o main.o -lfl
waylon@waylon-VirtualBox:/media/sf_ubuntu_share$ ./parser test
```

3. Get the result



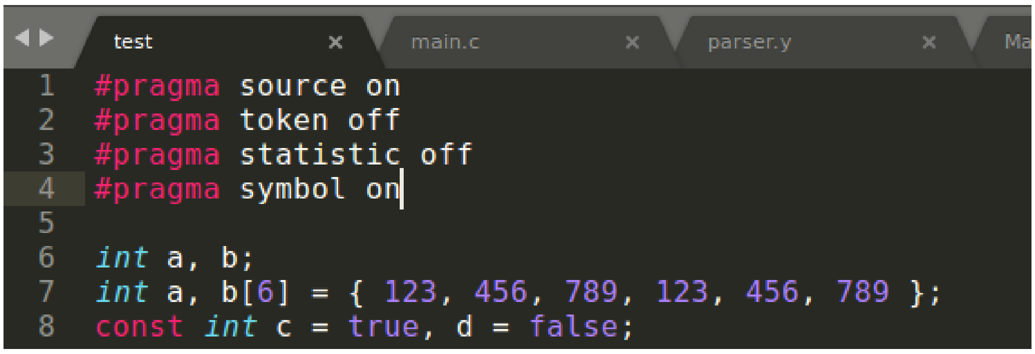
```
=======================================================================================
Name                             Kind       Level       Type               Attribute
---------------------------------------------------------------------------------------
a                                variable   0(global)   int
b                                variable   0(global)   int
c                                constant   0(global)   int                1
d                                constant   0(global)   int                0
e                                variable   0(global)   float
f                                variable   0(global)   float
h                                constant   0(global)   double             6.62607e-34
pi                               constant   0(global)   float              3.14159
s                                variable   0(global)   string
t                                constant   0(global)   string             this is a const string
flag                             variable   0(global)   bool[1][2][3]
test                             function   0(global)   int                int[2][2],int
main                             function   0(global)   int
funct                            function   0(global)   void               int[2][2],int[3]
nothing                          function   0(global)   void
======================================================================================

|--------------------------------|
|  There is no syntactic error!  |
|--------------------------------|
```

4. If the some semantic error is detected, the analyzer will print the error message and keep parsing.



```
##########Error at Line #23: assignment of constant variable c.##########
##########Error at Line #24: variable ar undeclared.##########
##########Error at Line #26: invalid access(2)##########
##########Error at Line #26: incompatible type for assignment.##########
##########Error at Line #27: invalid access(1)##########
##########Error at Line #30: incompatible type for assignment.##########
##########Error at Line #36: incompatible type for assignment.##########
##########Error at Line #39: invalid operands to operator '/'.##########
##########Error at Line #40: incompatible type for assignment.##########
##########Error at Line #41: incompatible type for assignment.##########
##########Error at Line #43: invalid operands to operator '=='.##########
##########Error at Line #44: invalid operands to operator '>'.##########
##########Error at Line #46: invalid operands to operator '!'.##########
##########Error at Line #47: variable z undeclared.##########
##########Error at Line #48: incompatible type for assignment.##########
##########Error at Line #53: invalid access(2)##########
##########Error at Line #53: invalid access(2)##########
##########Error at Line #53: incompatible type for assignment.##########
##########Error at Line #57: incompatible type for assignment.##########
##########Error at Line #60: incompatible type for argument 3.##########
##########Error at Line #62: too many arguments for the function.##########
##########Error at Line #63: too few arguments for the function.##########
##########Error at Line #64: too few arguments for the function.##########
##########Error at Line #65: too few arguments for the function.##########
```

5. Pragma is for compiler options. The symbol option enables printing symbol tables. All options are enabled by default.



```
#pragma source on
#pragma token off
#pragma statistic off
#pragma symbol on

int a, b;
int a, b[6] = { 123, 456, 789, 123, 456, 789 };
const int c = true, d = false;
```

Abilities:

With the previous version of scanner, parser and symbol table, this parser is able to point out semantic errors, such as redeclaration, type detection, type coercion and so on. The parser will keep working on finding as many semantic errors as possible.

Modifications:

1. Add type checking into the symbol table source file.
2. Check for semantic errors.
3. Modify the rule of function declaration and definitions.