ZJU C COMPILER

ZCC

ERROR HANDLING

AND ERROR RECOVERY

- 语法错误
  - 在建concrete parsing tree时找出错误并尽量恢复
- 语义错误
  - 在concrete parsing tree -> abstract parsing tree 的转换过程中生成

ERROR RECOVERY

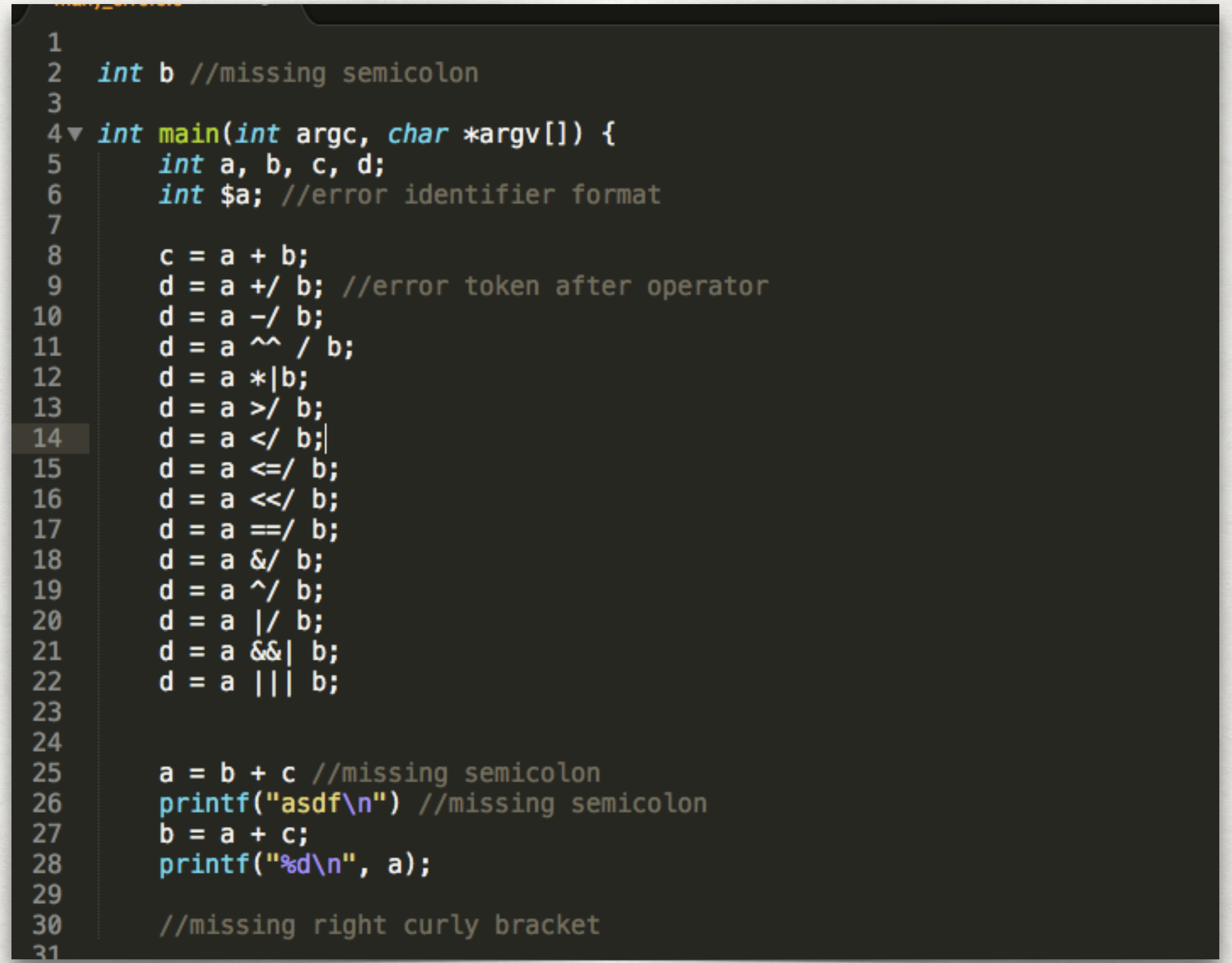
WHEN GENERATING THE PARSING TREE

- Adding error rules to our BNF
- Adding EOF token to handle the last missing right curly bracket
- Remove the error token from the parsing tree
- Insert the right but missing token into the parsing tree
- Using some counter to balance the curly brackets
- So that we can discover most common mistakes and do error recovery (still build the right parsing tree)

ERROR RECOVERY

WHEN GENERATING THE PARSING TREE

- Test file
- many_errors.c



```c

int b //missing semicolon

int main(int argc, char *argv[]) {
    int a, b, c, d;
    int $a; //error identifier format

    c = a + b;
    d = a +/ b; //error token after operator
    d = a -/ b;
    d = a ^^ / b;
    d = a *|b;
    d = a >/ b;
    d = a </ b;
    d = a <=/ b;
    d = a <</ b;
    d = a ==/ b;
    d = a &/ b;
    d = a ^/ b;
    d = a |/ b;
    d = a &&| b;
    d = a ||| b;


    a = b + c //missing semicolon
    printf("asdf\n") //missing semicolon
    b = a + c;
    printf("%d\n", a);

    //missing right curly bracket
```

ERROR HANDLING

WHEN CONCRETE TREE -> ABSTRACT TREE

- 1. 函数声明与函数定义的参数列表不一致
- 2. 变量重复定义
- 3. 赋值时 类型不匹配
- 4. 表达式中，操作数的类型与规定的类型不一致
- 5. typo，打字错误。
    - 使用edit distance
    - 从符号表中找出最接近的标识符，给出提示信息
- 6. 函数调用时参数表不符合函数定义
- 7. 函数实际返回值类型 不符合 函数定义中的函数返回值类型

基本功能

- 计算add sub mul div 等
- 逻辑and or not
- 跳转jmp je jg jl
- 移位sal sar
- 函数call ret
- 堆栈push pop
- 全局数据 常量浮点数、字符串、global、static变量
- 浮点数运算fld fstp fadd fsub fmul fdiv

优化

- 前端：
  - constant folding
  - 死代码消除
- 后端优化：
  - 寄存器优化：
    - 将ebx,ecx,edx作为临时变量的暂存区域
    - 将esi edi作为eax的交换区
  - 指令优化：
    - *2 / 4 / 8。。。 ->sal
    - lea 2*eax+offset -> reg

支持特性