编译器实验报告

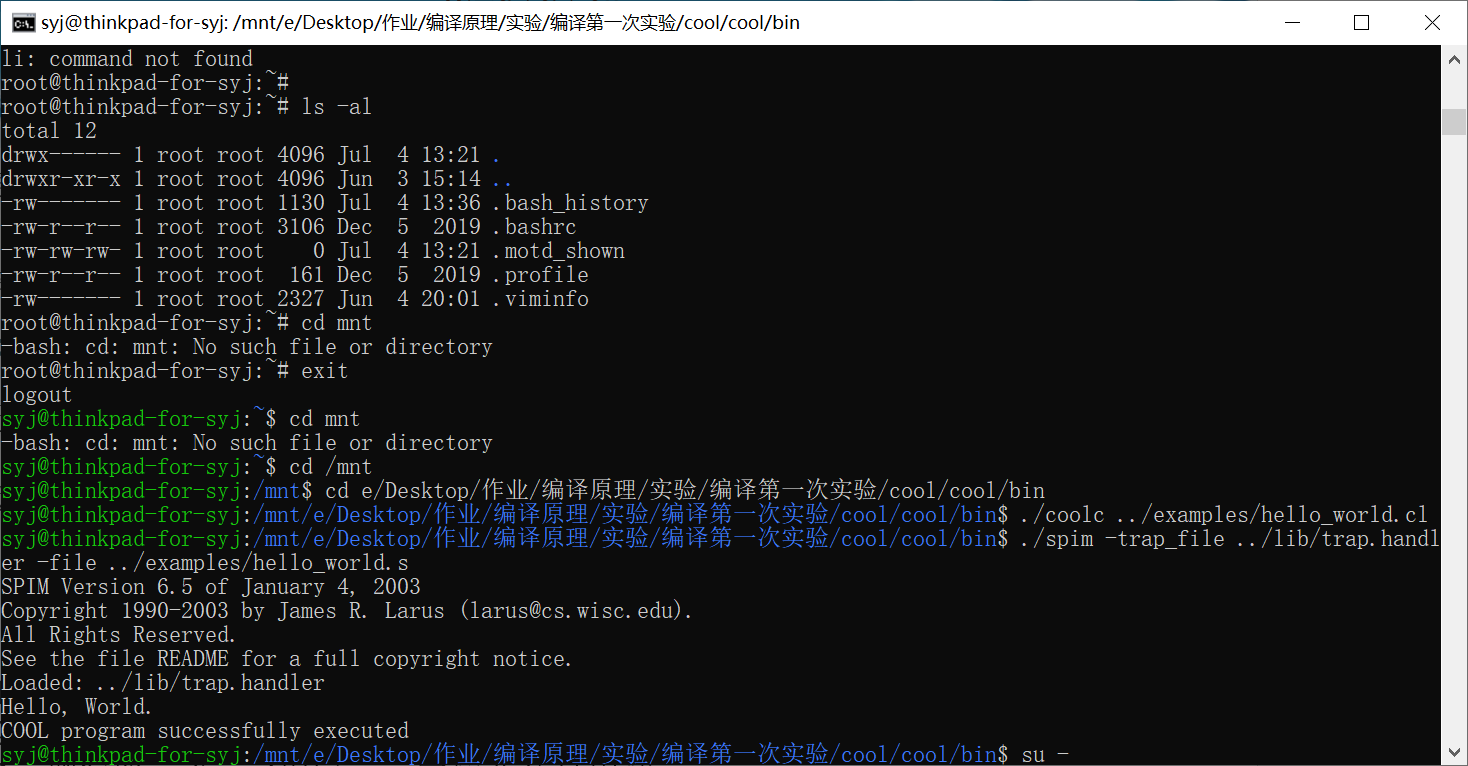
计算机82班 施炎江

主机名：thinkpad-for-syj

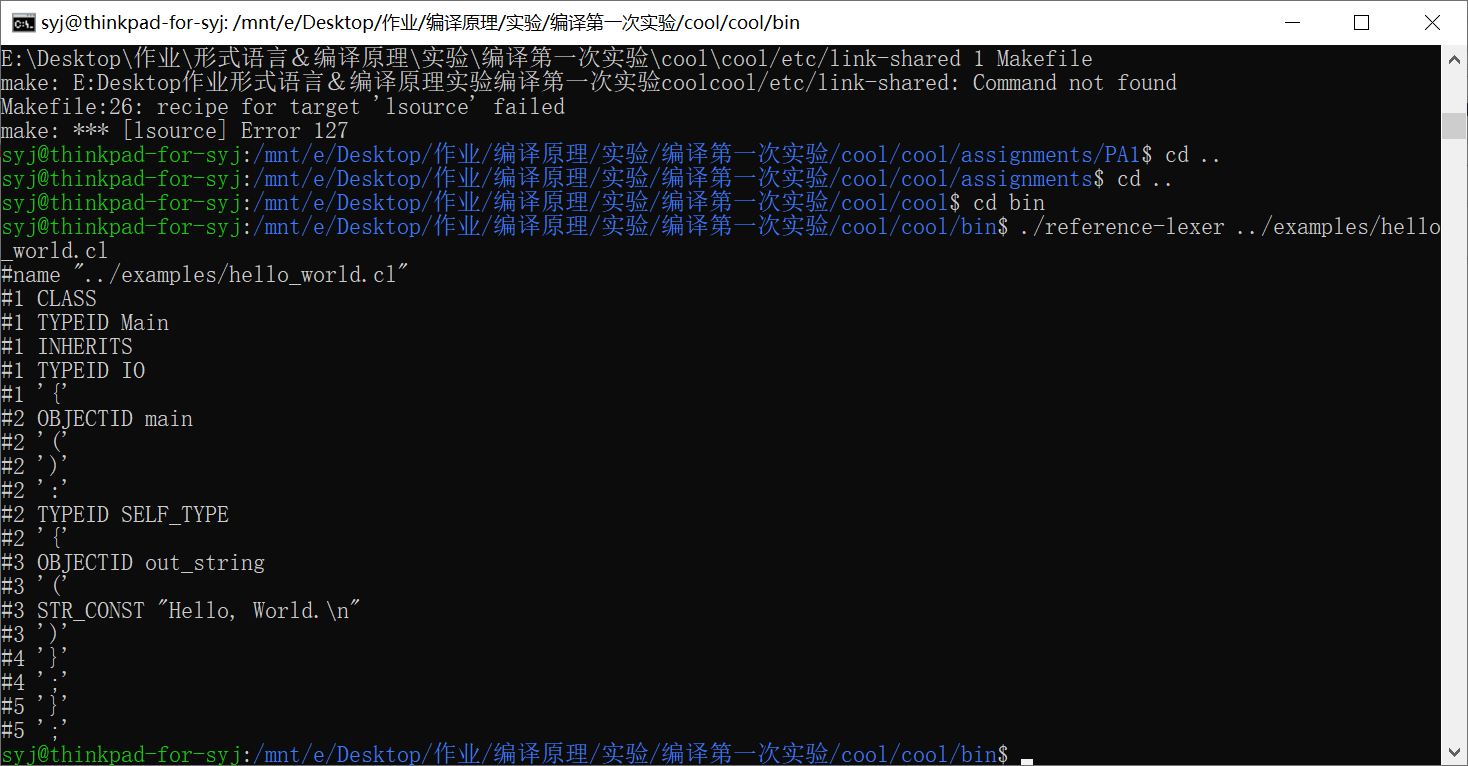
用户名：syj

# 第一次实验

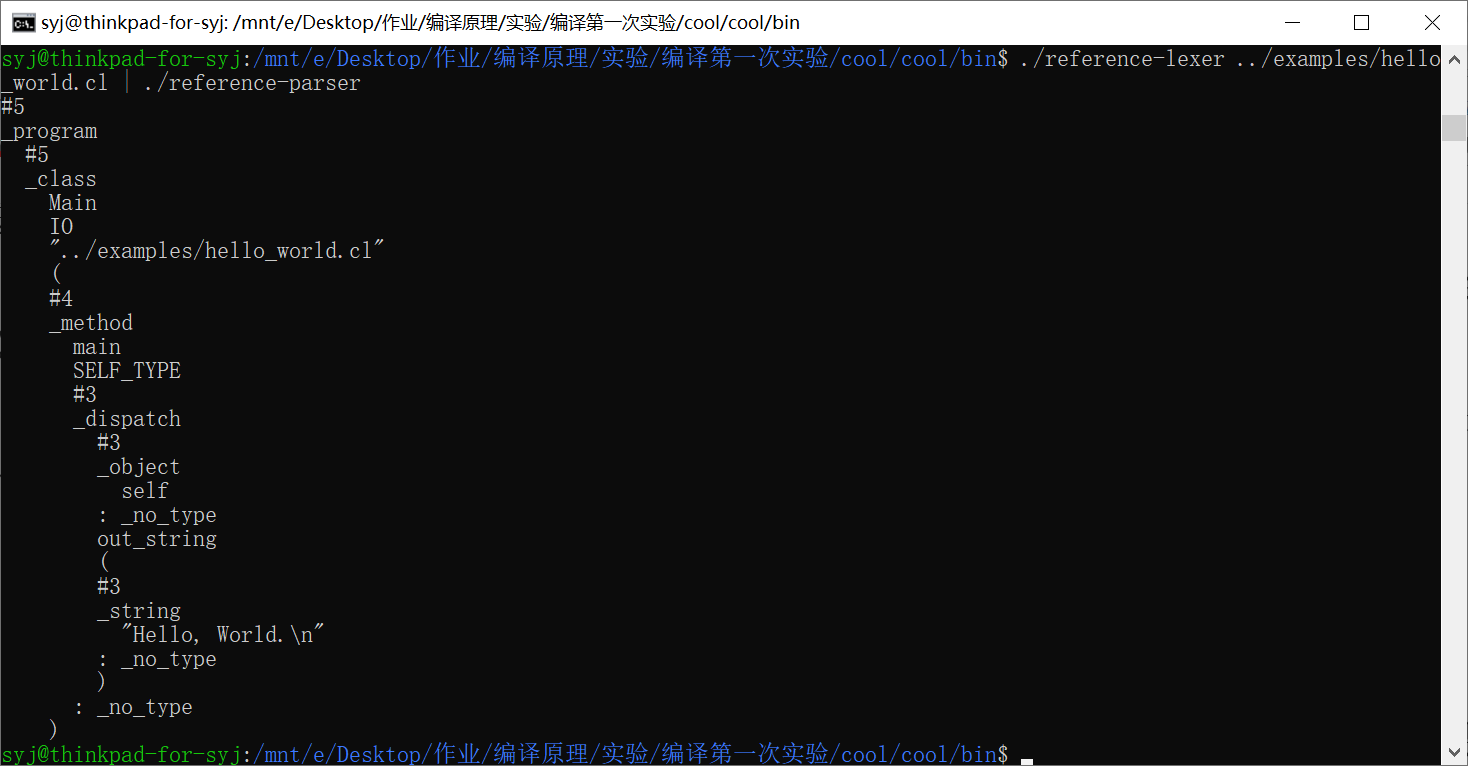
1. 编译运行hello world



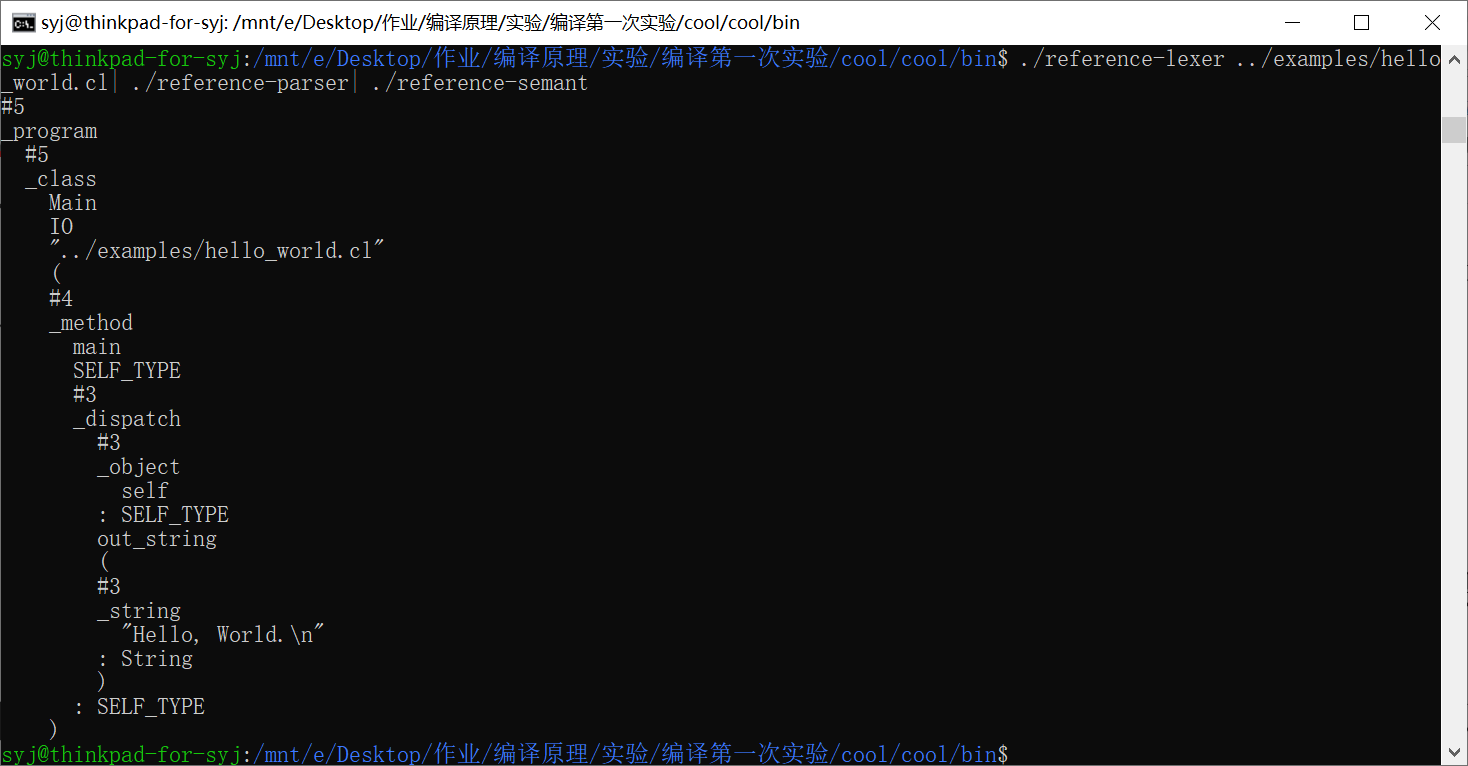
1. 用标准的词法分析程序，编译一个Cool语言程序，观察结果



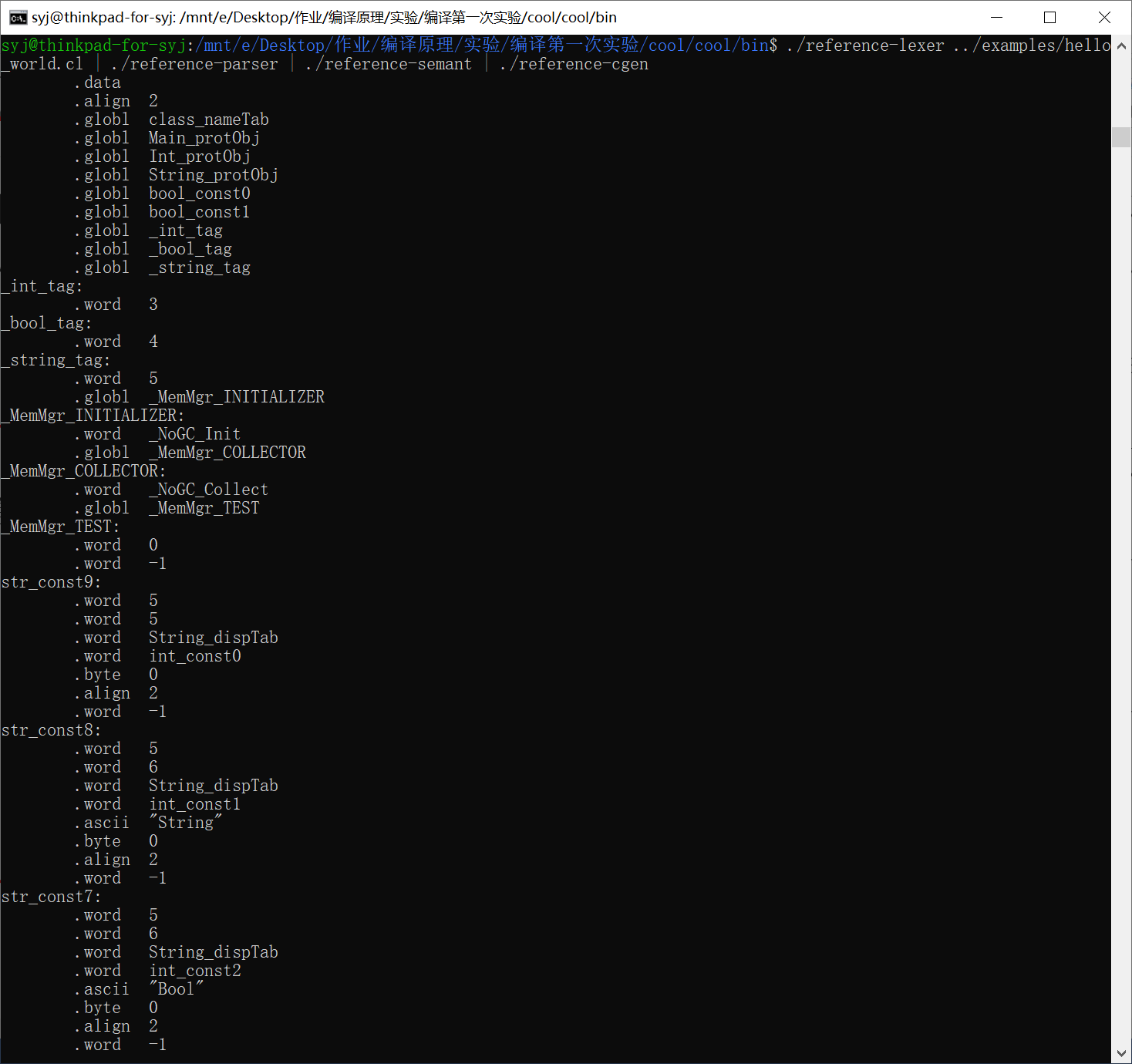
1. 用标准的词法分析程序和语法分析程序，编译一个Cool语言程序，观察结果



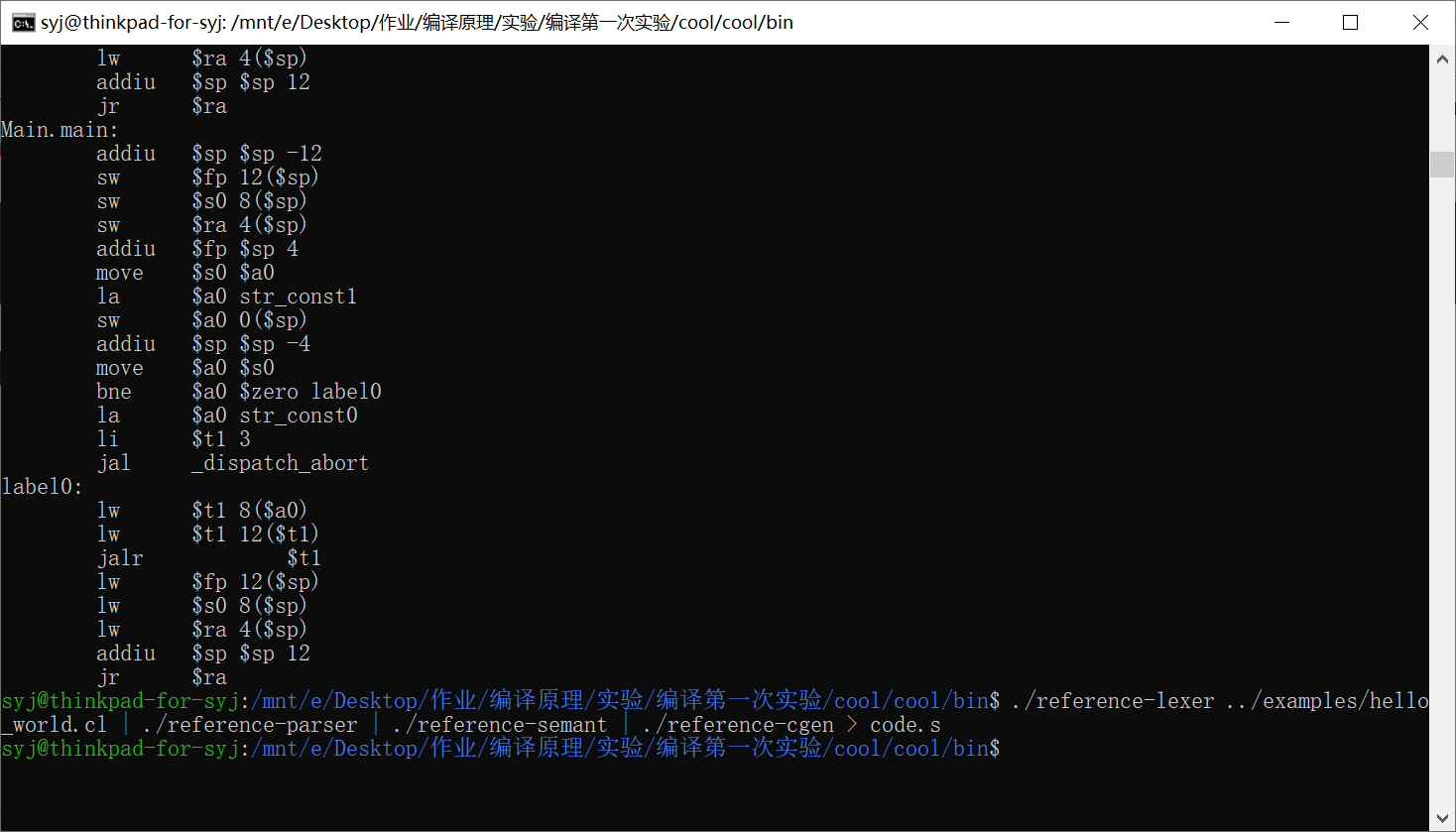
1. 用标准的词法分析程序，语法分析程序和语义分析程序编译一个Cool语言程序，观察结果



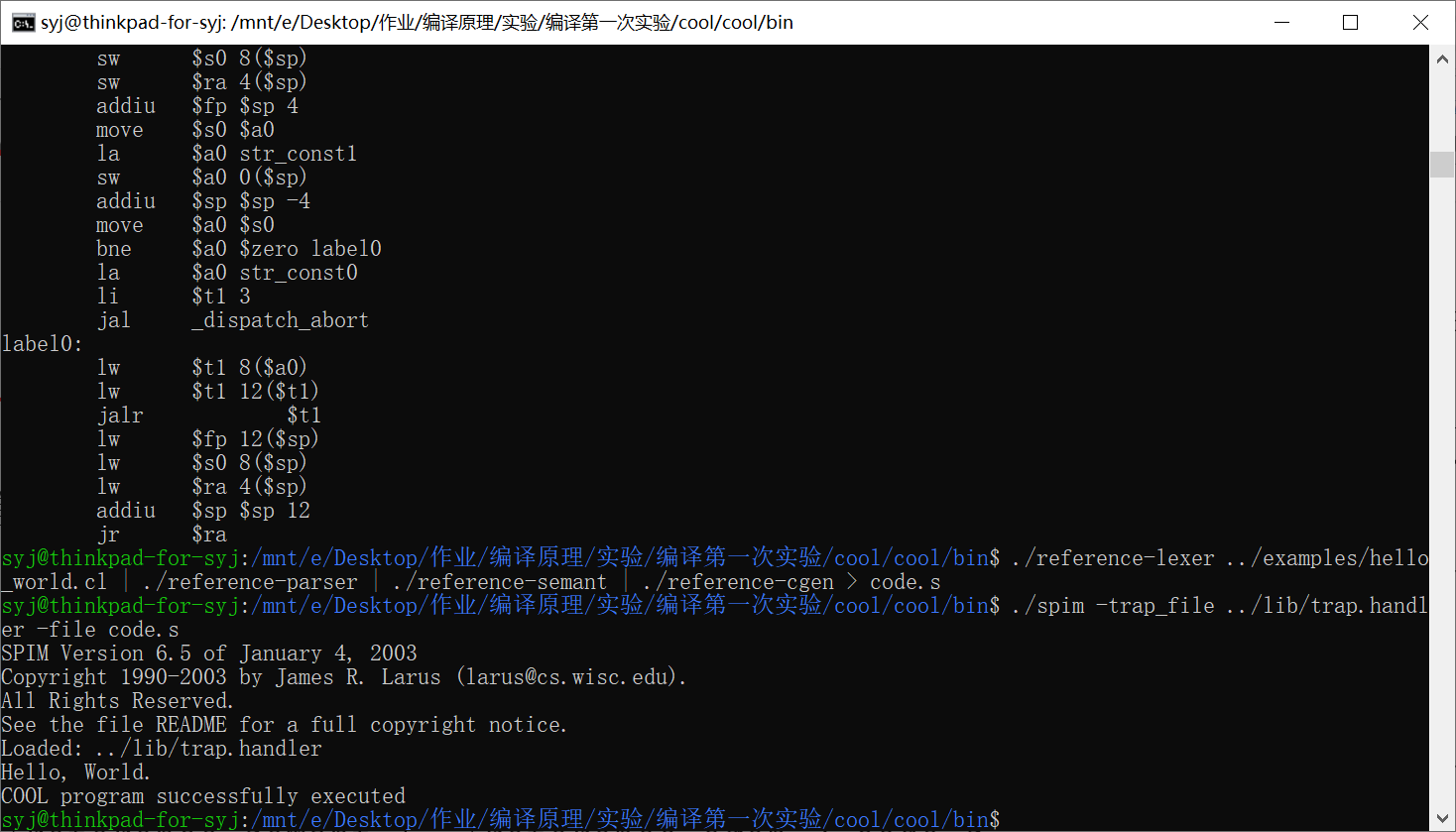
1. 用标准的词法分析程序，语法分析程序，语义分析程序和代码生成程序共同编译Cool语言程序，生成最终的汇编代码（由于输出较多，仅截取部分输出结果示意）



1. 将得到的汇编代码输出到.s文件中



1. 将生成的汇编代码在spim上运行



# 第二次实验

1. Stack.cl代码

(\*

   The class A2I provides integer-to-string and string-to-integer

conversion routines.  To use these routines, either inherit them

in the *class* where needed, have a dummy variable bound to

something of type A2I, or simpl write (new A2I).method(argument).

\*)

(\*

   c2i   Converts a 1-character string to an integer.  Aborts

         if the string is not "0" through "9"

\*)

*class* A2I {

     c2i(*char* : String) : Int {

        if *char* = "0" then 0 else

        if *char* = "1" then 1 else

        if *char* = "2" then 2 else

        if *char* = "3" then 3 else

        if *char* = "4" then 4 else

        if *char* = "5" then 5 else

        if *char* = "6" then 6 else

        if *char* = "7" then 7 else

        if *char* = "8" then 8 else

        if *char* = "9" then 9 else

        { abort(); 0; }  -- the 0 is needed to satisfy the typchecker

        fi fi fi fi fi fi fi fi fi fi

     };

(\*

   i2c is the inverse of c2i.

\*)

     i2c(i : Int) : String {

        if i = 0 then "0" else

        if i = 1 then "1" else

        if i = 2 then "2" else

        if i = 3 then "3" else

        if i = 4 then "4" else

        if i = 5 then "5" else

        if i = 6 then "6" else

        if i = 7 then "7" else

        if i = 8 then "8" else

        if i = 9 then "9" else

        { abort(); ""; }  -- the "" is needed to satisfy the typchecker

        fi fi fi fi fi fi fi fi fi fi

     };

(\*

   a2i converts an ASCII string into an integer.  The empty string

is converted to 0.  Signed and *unsigned* strings are handled.  The

method aborts if the string does not represent an integer.  Very

*long* strings of digits produce strange answers because of arithmetic

overflow.

\*)

     a2i(s : String) : Int {

        if s.length() = 0 then 0 else

        if s.substr(0,1) = "-" then ~a2i\_aux(s.substr(1,s.length()-1)) else

        if s.substr(0,1) = "+" then a2i\_aux(s.substr(1,s.length()-1)) else

           a2i\_aux(s)

        fi fi fi

     };

(\*

  a2i\_aux converts the usigned portion of the string.  As a programming

example, this method is written iteratively.

\*)

     a2i\_aux(s : String) : Int {

        (let *int* : Int <- 0 in

           {

               (let j : Int <- s.length() in

                  (let i : Int <- 0 in

                    while i < j loop

                        {

*int* <- *int* \* 10 + c2i(s.substr(i,1));

                            i <- i + 1;

                        }

                    pool

                  )

               );

*int*;

            }

        )

     };

(\*

    i2a converts an integer to a string.  Positive and negative

numbers are handled correctly.

\*)

    i2a(i : Int) : String {

        if i = 0 then "0" else

if 0 < i then i2a\_aux(i) else

          "-".concat(i2a\_aux(i \* ~1))

        fi fi

    };

(\*

    i2a\_aux is an example using recursion.

\*)

    i2a\_aux(i : Int) : String {

        if i = 0 then "" else

            (let next : Int <- i / 10 in

                i2a\_aux(next).concat(i2c(i - next \* 10))

            )

        fi

    };

};

*class* List inherits IO

{

        isNil() : Bool

        {

                {

                --out\_string("list\n");

                        true;

                }

        };

        head() : String

        {

                {

                        abort();

                        "";

                }

        };

        tail() : List

        {

                {

abort();

                        self;

                }

        };

        cons(i : String) : List

        {

                (new Cons).init(i, self)

        };

};

*class* Cons inherits List

{

        first : String;

        rest :  List;

        isNil() : Bool

        {

                {

                        --out\_string("cons\n");

                        false;

                }

        };

        head() : String

        {

                first

        };

        tail() : List

        {

                rest

        };

        init(head : String, next : List) : List

        {

                {

                        first <- head;

                        rest  <- next;

                        self;

}

        };

};

*class* Main inherits IO

{

        stack : List;

        newline() : Object

        {

                out\_string("\n")

        };

        prompt() : String

        {

                {

                        out\_string(">");

                        in\_string();

                }

        };

        display\_stack(s : List) : Object

        {

                {

                        --out\_string("hello\n");

                        if s.isNil() then out\_string("")

                                        else

                                        {

                                                out\_string(s.head());

                                                out\_string("\n");

                                                display\_stack(s.tail());

                                        }

                fi;

                }

        };

        main():Object

        {

                ( let z : A2I <- new A2I  , stack : List <- new List in

                        while true loop

                        ( let s : String <- prompt() in

                                if s = "x" then

                                        abort()

                                else

                                        if s = "d"  then

                                                display\_stack(stack)

                                        else

                                            if s = "e"  then

                                                        {

                                                                if stack.isNil() then out\_string("")

                                                                else

                                                                if stack.head() = "+" then

                                                                        {

                                                                                stack <- stack.tail();

                                                                                (let a : Int <- new Int, b : Int <- new Int in

                                                                                        {

                                                                                                --out\_string(stack.head());

                                                                                                a  <- z.a2i(stack.head());

                                                                                                stack <- stack.tail();

                                                                                                b  <- z.a2i(stack.head());

                                                                                                stack <- stack.tail();

                                                                                                a <- a + b;

                                                                                                --out\_string(z.i2a(a));

                                                                                                stack <- stack.cons(z.i2a(a));

                                                                                        }

                                                                                );

                                                                        }

                                                                else

                                                                        if stack.head() = "s" then

                                                                                {

                                                                                        stack <- stack.tail();

                                                                                        (let a : String <- new String , b : String <- new String in

                                                                                                {

                                                                                                        a  <- stack.head();

                                                                                                        stack <- stack.tail();

                                                                                                        b  <- stack.head();

                                                                                                        stack <- stack.tail();

                                                                                                        stack <- stack.cons(a);

                                                                                                        stack <- stack.cons(b);

                                                                                                }

                                                                                        );

                                                                                }

           else

                          out\_string("")

                                                                        fi

                                                                        fi

                                                                fi;

                                                        }

                                                else

                                                        stack <- stack.cons(s)

                                                fi

                                        fi

                                fi

                        )

                        pool

                )

        };

# };第三次实验：

1. 统计字符个数，单词个数，行数

%{

*int* num\_char=0;

*int* num\_line=0;

*int* num\_word=0;

%}

%%

[ \*t*]

\n {num\_line++;}

[^ \t\*n*]+ {num\_char+=yyleng; num\_word++;}

%%

*int* yywrap(){}

*int* main(){

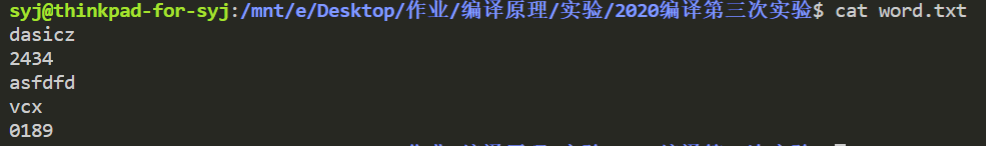
    yylex();

    printf("num\_char=%d\nnum\_word=%d\nnum\_line=%d\n",num\_char,num\_word,num\_line);

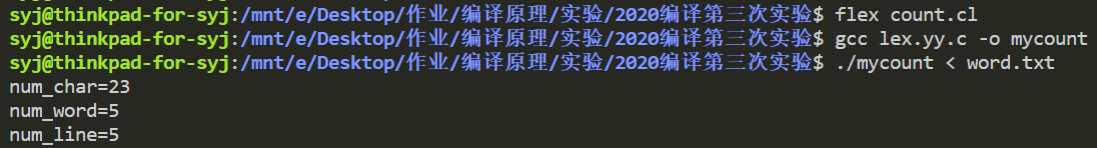
    return 0;

}

被统计文件



运行结果



1. 统计if语句

%{

*int* num\_if=0;

%}

%%

(if) {num\_if++;}

%%

*int* yywrap(){}

*int* main (){

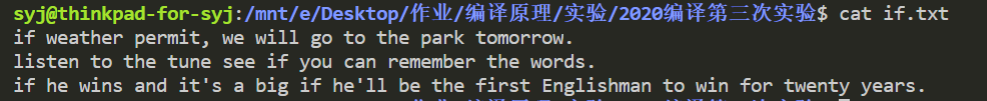
yylex();

printf("num\_ if=%d\n",num\_if);

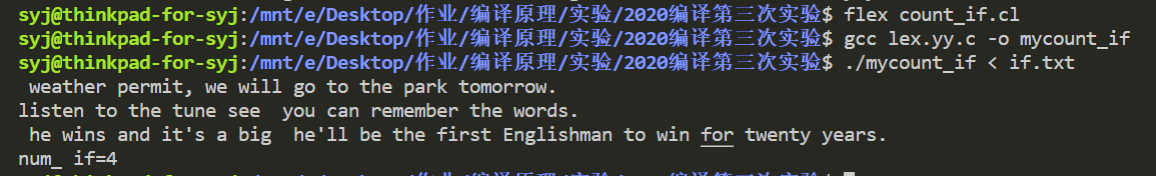
return 0;

}

被统计文件



运行结果



1. 多重入口

%{

#include<stdio. h>

%}

%start AA BB CC

%%

^a {ECHO; BEGIN AA; }

^b {ECHO; BEGIN BB; }

^C { ECHO; BEGIN CC; }

\n|(\t)+|"'

"+ {ECHO;BEGIN 0;}

<AA>magic {printf("first\n"); }

<BB>magic {pr intf(" second\n");}

<CC>magic {printf("third\n");}

magic {printf(" zero\n");}

%%

*int* yywrap(){}

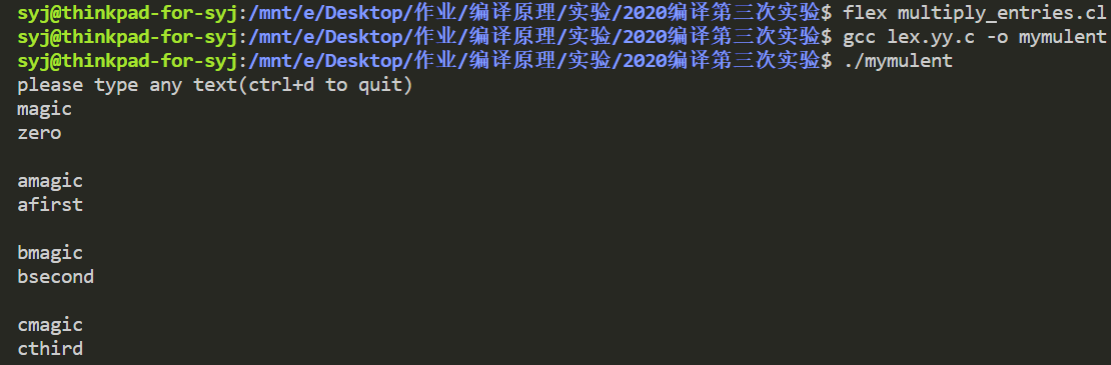
*int* main ( ){

pr intf("please tpe any text(ctrl+d to quit)\n");

yyLex();

return ;

运行结果



# 第四次实验

1. cool.flex文件

/\*

 \*  这个文件用来生成一个COOL语言的词法分析程序.

 \*/

/\*

 \*  lex文件的第一个部分，也就是包含在“%{”和“ %}”之间的部分，是用来像未来的词法分析程序输出代

 \*  码的，也就是说这里的需要include头文件，extern外部变量，因为这部分是要直接照搬到以后的.c文件中去的

 \*/

%{

#include <cool-parse.h>  //记号的定义放在cool-parse.h文件中

#include <stringtab.h>

#include <utilities.h>

/\* 词法分析程序需要的宏定义 \*/

#define yylval cool\_yylval

#define yylex  cool\_yylex

/\* 字符串常量的最大长度 \*/

#define MAX\_STR\_CONST 1025

#define YY\_NO\_UNPUT   /\* 让g++的编译结果变得友好 \*/

extern FILE \*fin; /\* 从这个文件指针读取记号 \*/

/\* 定义YY\_INPUT以后我们就可以从fin中读取记号了:

 \*/

#undef YY\_INPUT

#define YY\_INPUT(*buf*,*result*,*max\_size*) \

   if ( (result = fread( (*char*\*)buf, sizeof(*char*), max\_size, fin)) < 0) \

      YY\_FATAL\_ERROR( "read() in flex scanner failed");

*char* string\_buf[MAX\_STR\_CONST]; /\* 记录字符串的字符数组\*/

*char* \*string\_buf\_ptr;

extern *int* curr\_lineno;

extern *int* verbose\_flag;

extern YYSTYPE cool\_yylval;

/\*

 \*  在这里添加你自己的头文件和变量

 \*/

*int* uniqueIndex =1;

%}

/\*

 \* 第二部分用来定义正则表达式需要的“元素”

 \*/

/\* 下面是我们给出的基本实例 \*/

TYPEID          [A-Z]+[\_A-Za-z0-9]\*

OBJECTID        [a-z]+[\_a-zA-Z0-9]\*

STR\_CONST       \".\*\"

INT\_CONST       [0-9]+

WHITE           [ \t]+

LINE            \n

%s              MutiCom

%%/\*第二部分结束\*/

"--"[^\n]\*        {/\* 忽略掉一行的注释 \*/}

"(\*"              {BEGIN(MutiCom);}

<MutiCom>[^\n\*]\*   {}

<MutiCom>"\*"+[^\*)\n]\* {}

<MutiCom>\n       {++curr\_lineno;}

<MutiCom>"\*"+")"  {BEGIN(INITIAL);}

"class"           {return CLASS;}

"inherits"        {return INHERITS;}

"if"              {return IF;}

"else"            {return ELSE;}

"then"            {return THEN;}

"fi"              {return FI;}

"let"             {return LET;}

"in"              {return IN;}

"while"           {return WHILE;}

"loop"            {return LOOP;}

"pool"            {return POOL;}

"case"            {return CASE;}

"esac"            {return ESAC;}

"of"              {return OF;}

"new"             {return NEW;}

"isvoid"          {return ISVOID;}

"true"            {

                     cool\_yylval.boolean = 1;

                     return BOOL\_CONST;

                  }

"false"           {

                     cool\_yylval.boolean=0;

                     return BOOL\_CONST;

                  }

"not"             {return NOT;}

"SELF\_TYPE"       {

                     cool\_yylval.symbol=new IdEntry(yytext,strlen(yytext),uniqueIndex++);

                     return TYPEID;

                  }

"self"            {

                     cool\_yylval.symbol=new IdEntry(yytext,strlen(yytext),uniqueIndex++);

                     return OBJECTID;

                  }

"le"              {return LE;}

"<-"              {return ASSIGN;}

"=>"              {return DARROW;}

"{"               {return '{';}

"}"               {return '}';}

"("               {return '(';}

")"               {return ')';}

":"               {return ':';}

";"               {return ';';}

"+"               {return '+';}

"-"               {return '-';}

"\*"               {return '\*';}

"/"               {return '/';}

"="               {return '=';}

"<"               {return '<';}

"."               {return '.';}

"~"               {return '~';}

","               {return ',';}

"@"               {return '@';}

{TYPEID}          {

                    cool\_yylval.symbol = new IdEntry(yytext,strlen(yytext),uniqueIndex++);

                    return TYPEID;

                  }

{OBJECTID}        {

                    cool\_yylval.symbol = new IdEntry(yytext,strlen(yytext),uniqueIndex++);

                    return OBJECTID;

                  }

{STR\_CONST}       {

                    cool\_yylval.symbol = new StringEntry(yytext,strlen(yytext),uniqueIndex++);

                    return STR\_CONST;

                  }

{INT\_CONST}       {

                    cool\_yylval.symbol = new IntEntry(yytext,strlen(yytext),uniqueIndex++);

                    return INT\_CONST;

                  }

{WHITE}           {}

{LINE}            {curr\_lineno++;}

.                {

                    cool\_yylval.error\_msg = new *char*[strlen(yytext+1)];

                    strcpy(cool\_yylval.error\_msg,yytext);

                    return ERROR;

                  }

%%

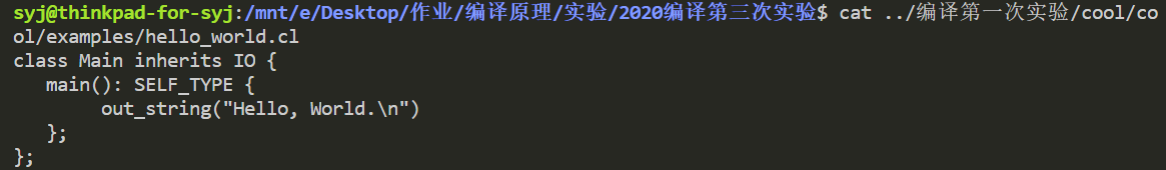
int yywrap()

{

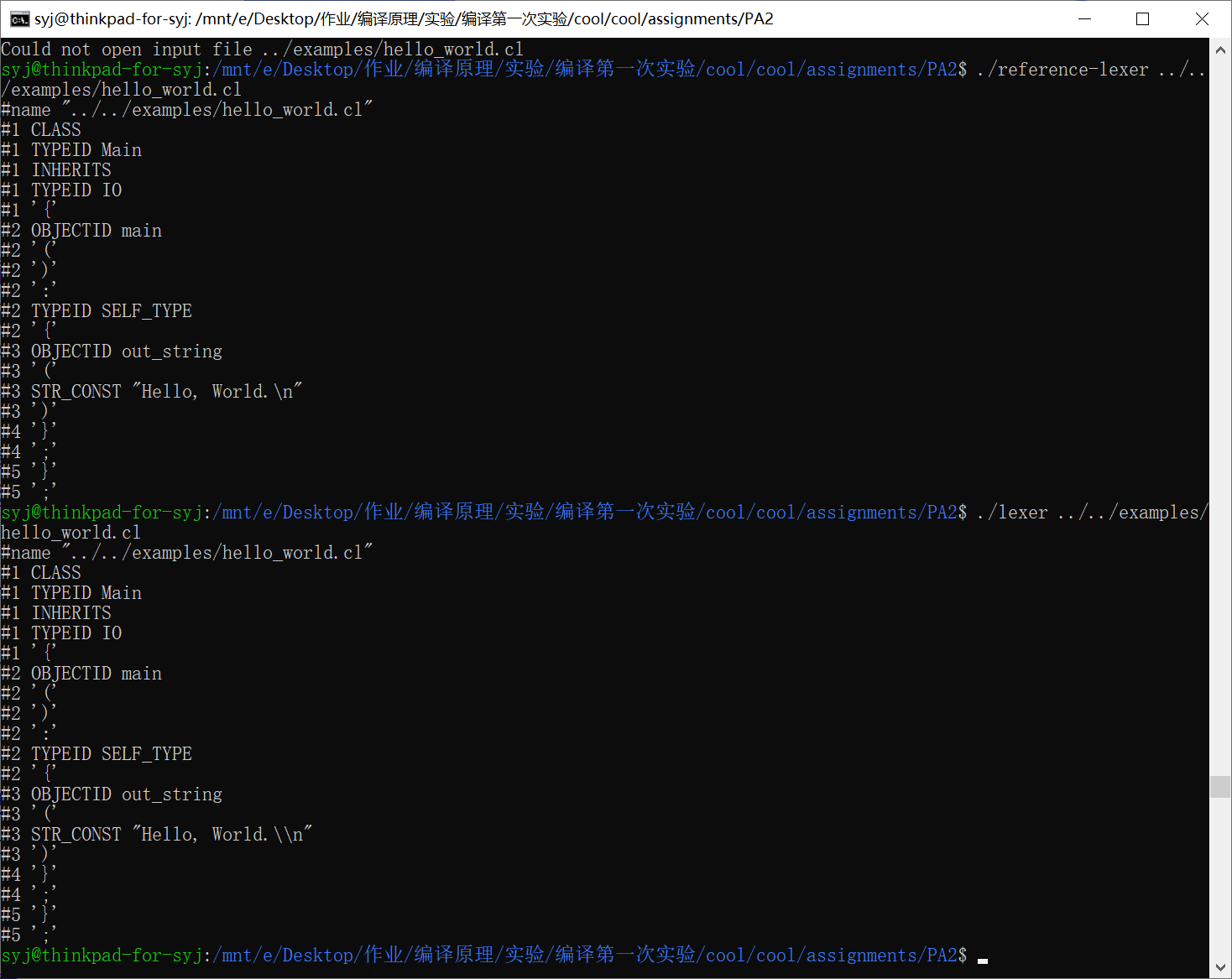
   return 1;

}

Hello\_world.cl文件



1. 使用make指令生成的lexer和标准词法分析器分别对hello\_world.cl进行词法分析



经对比发现，二者输出结果相同，都对目标代码进行了词法分析

# 第五次实验

1. 简易计算器

token.l文件

%{

#include "y.tab.h"

%}

%%

[0-9]+ {yylval=atoi(yytext); return T\_NUM;}

[-/+\*()\n] {return yytext[0];}

. {return 0;}

%%

*int* yywrap(){

return 1;

}

parser.y文件

%{

#include<stdio.h>

extern *int* yylex();

extern *int* yyparse();

*void* yyerror(const *char*\* *msg*){}

%}

%token T\_NUM

%left '+' '-'

%left '\*' '/'

%%

S: S E '\n' {printf("ans=%d\n",$2);}

 |          {  }

 ;

E: E '+' E {$$=$1+$3;}

 | E '-' E {$$=$1-$3;}

 | E '\*' E {$$=$1\*$3;}

 | E '/' E {$$=$1/$3;}

 | T\_NUM {$$=$1;}

 | '(' E ')' {$$=$2;}

 ;

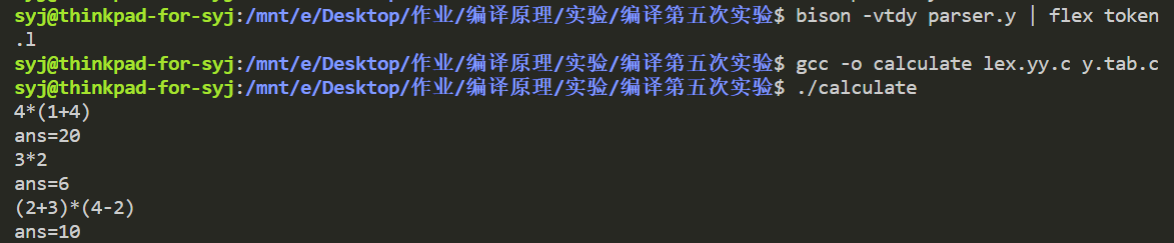
%%

*int* main (){

return yyparse();

}

运行结果



1. 布尔表达式

token.l文件

%{

#include"y.tab.h"

%}

%%

true {yylval=1;return T\_BOOL;}

false {yylval=0;return T\_BOOL;}

&& {return T\_AND;}

\|\| {return T\_OR;}

! {return T\_NOT;}

\( {return T\_LS;}

\) {return T\_RS;}

\n {return T\_NEWLINE;}

. {return 0;}

%%

*int* yywrap(){

return 1;

}

parser.y文件

%{

#include<stdio.h>

extern *int* yylex();

extern *int* yyparse( );

*void* yyerror(const *char*\* *msg*){};

%}

%token T\_BOOL T\_AND T\_OR T\_NOT T\_LS T\_RS T\_NEWLINE

%%

S : E T\_NEWLINE {printf("ans=%s\n",$1?"true":"false");}

  |             {  }

  ;

E : T T\_OR T {$$=$1||$3;}

  | T {$$=$1;}

  ;

T : F T\_AND F {$$=$1&&$3;}

  | F {$$=$1;}

  ;

F : T\_LS E T\_RS {$$=$2;}

  | T\_NOT F {$$=!$2;}

  | T\_BOOL {$$=$1;}

  ;

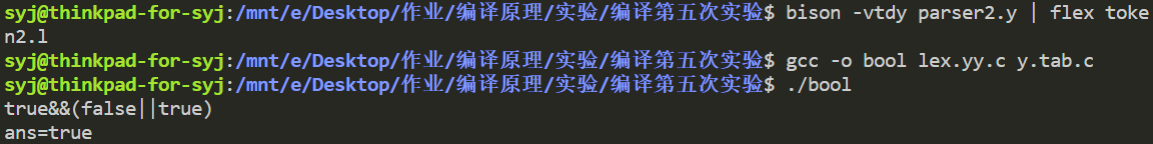
%%

*int* main () {

return yyparse();

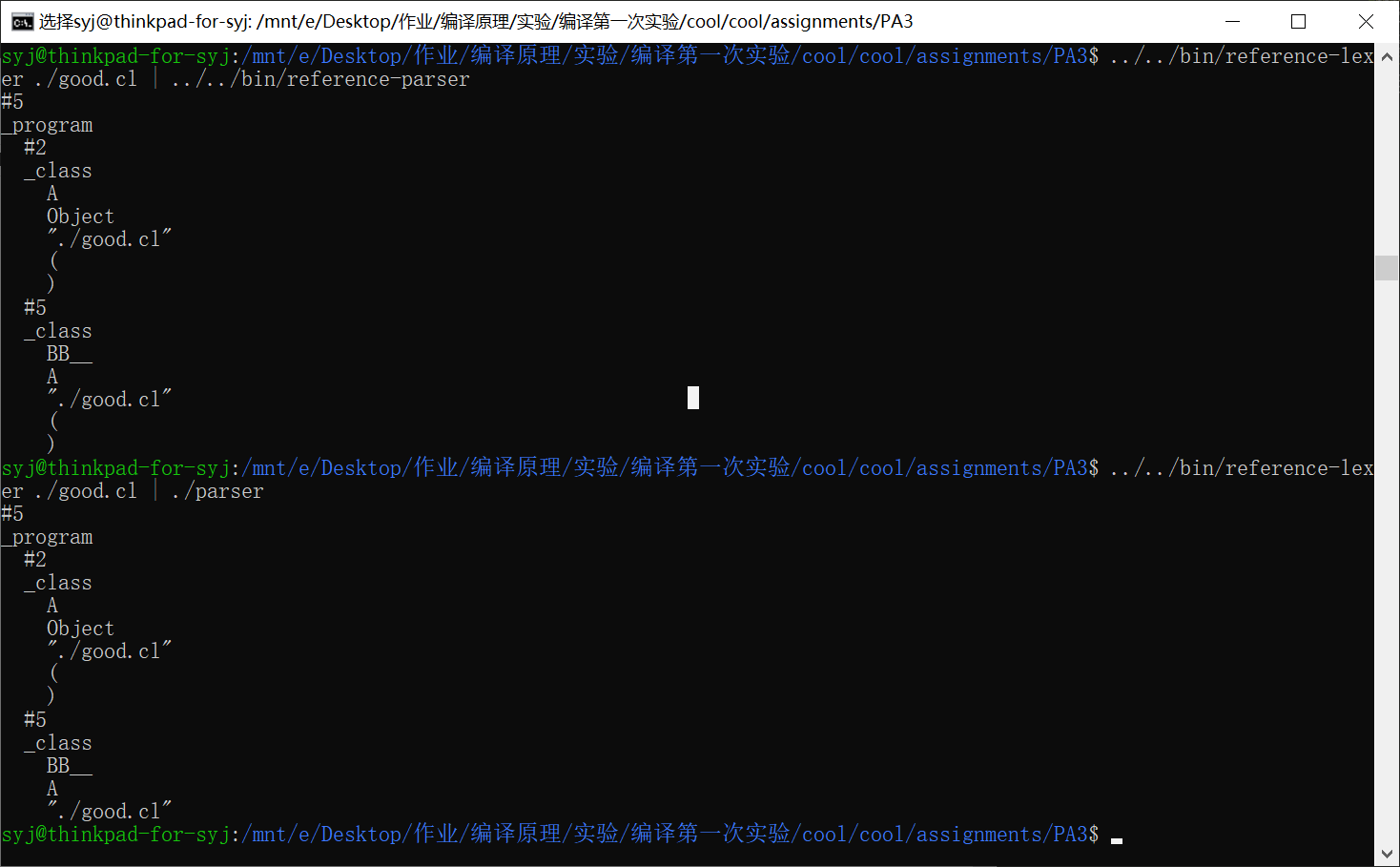
}

运行结果



# 第六次实验

1. 先用lexer对good.cl进行词法分析，然后分别用make生成的语法分析器和标准语法分析器对结果进行语法分析



对比结果发现，二者输出相同。