南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

CS323 Lab 7

Yepang Liu
liuyp1@sustech.edu.cn

Agenda

- Bison exercises
  - Modify the calculator example to support parentheses
  - Validate IP address

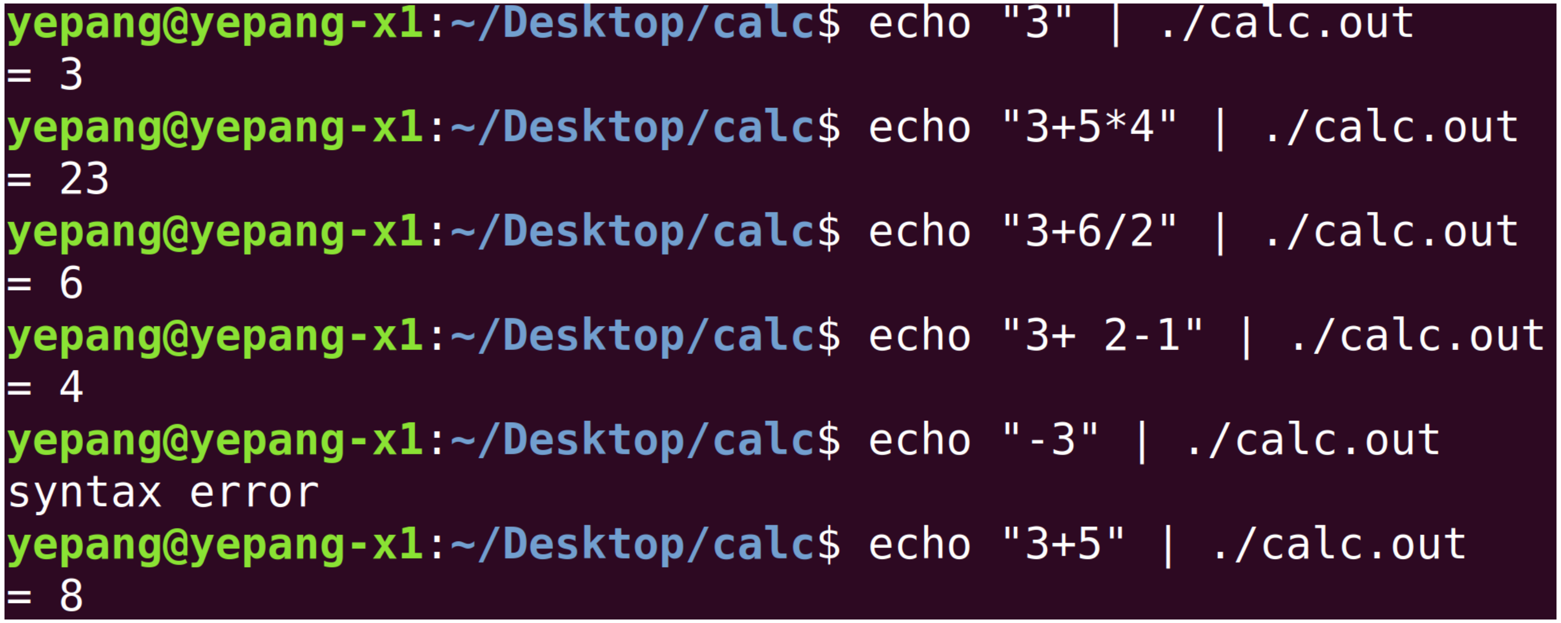
Structure of YACC Source Programs

- **Declarations (声明)**
  - Ordinary C declarations
  - Grammar tokens
- **Translation rules (翻译规则)**
  - Rule = a production + semantic action
- **Supporting C routines (辅助性C语言例程)**
  - Directly copied to `y.tab.c`
  - Can be invoked in the semantic actions
  - Other procedures such as error recovery routines may be provided

```
declarations
%%
translation rules
%%
supporting C routines
```

The calculator Demo in Lab #6

```
Calc -> Exp
Exp -> Factor | Exp ADD Factor | Exp SUB Factor
Factor -> Term | Factor MUL Term | Factor DIV Term
Term -> INT
```



```
yepang@yepang-x1:~/Desktop/calc$ echo "3" | ./calc.out
= 3
yepang@yepang-x1:~/Desktop/calc$ echo "3+5*4" | ./calc.out
= 23
yepang@yepang-x1:~/Desktop/calc$ echo "3+6/2" | ./calc.out
= 6
yepang@yepang-x1:~/Desktop/calc$ echo "3+ 2-1" | ./calc.out
= 4
yepang@yepang-x1:~/Desktop/calc$ echo "-3" | ./calc.out
syntax error
yepang@yepang-x1:~/Desktop/calc$ echo "3+5" | ./calc.out
= 8
```

Flex/Bison Code for Calculator

lex.l

```
%{
    #include "syntax.tab.h"
    #include "stdlib.h"
%}
%%
[0-9]+ { yylval = atoi(yytext); return INT; }
"+" { return ADD; }
"-" { return SUB; }
"*" { return MUL; }
"/" { return DIV; }
[ \n\r\t] {}
. { fprintf(stderr, "unknown symbol: %s\n", yytext);
    exit(1); }
```

syntax.y

```
%{
    #include "lex.yy.c"
    void yyerror(const char*);
%}
%token INT
%token ADD SUB MUL DIV
%%
Calc: /* to allow empty input */
    | Exp { printf("= %d\n", $1); }
    ;
Exp: Factor
    | Exp ADD Factor { $$ = $1 + $3; }
    | Exp SUB Factor { $$ = $1 - $3; }
    ;
Factor: Term
    | Factor MUL Term { $$ = $1 * $3; }
    | Factor DIV Term { $$ = $1 / $3; }
Term: INT
    ;
%%
void yyerror(const char *s) {
    fprintf(stderr, "%s\n", s);
}
int main() {
    yyparse(); // will invoke yylex()
}
```

yylval:

- Flex internal variable that is used to store the attribute of a recognized token
- Its data type is YYSTYPE (int by default)*
- After storing values to `yylval` in Flex code, the values will be propagated to Bison (i.e., the syntax anlayzer part) and can be retrieved using `$n`

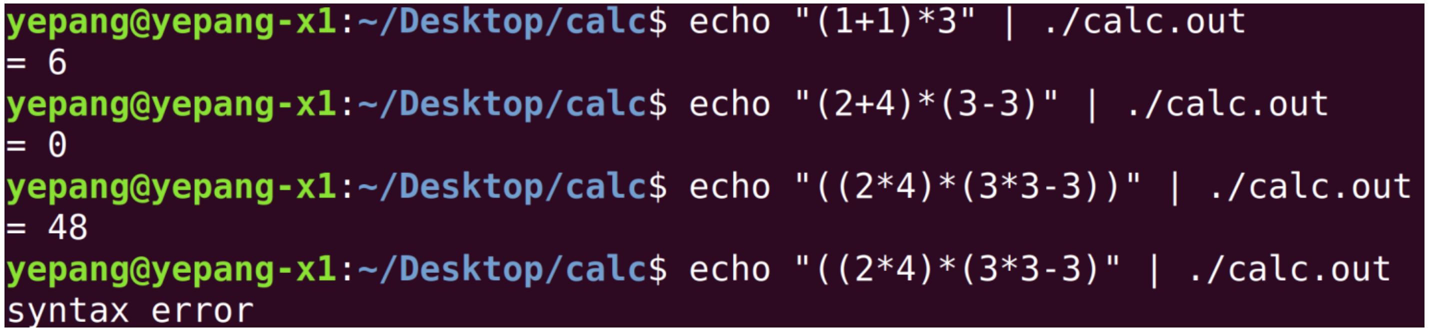
*Can be customized by putting command like `#define YYSTYPE char*` at the beginning of .l and .y files.

Lab Exercise 1

- Modify the `lex.l` and `syntax.y` to support parentheses

```
Calc -> Exp
Exp -> Factor | Exp ADD Factor | Exp SUB Factor
Factor -> Term | Factor MUL Term | Factor DIV Term
Term -> LP Exp RP | INT
```

*Red for non-terminals, Blue for terminals, Calc is the start symbol



```
yepang@yepang-x1:~/Desktop/calc$ echo "(1+1)*3" | ./calc.out
= 6
yepang@yepang-x1:~/Desktop/calc$ echo "(2+4)*(3-3)" | ./calc.out
= 0
yepang@yepang-x1:~/Desktop/calc$ echo "((2*4)*(3*3-3))" | ./calc.out
= 48
yepang@yepang-x1:~/Desktop/calc$ echo "((2*4)*(3*3-3)" | ./calc.out
syntax error
```

Validate IP Address (leetcode #468)

- Use Bison and Flex to complete the following task:
  - Given a string *queryIP*, output "IPv4" if *queryIP* is a valid IPv4 address, "IPv6" if *queryIP* is a valid IPv6 address or "Invalid" otherwise

- A valid IPv4 address is an IP in the form of "$x_1.x_2.x_3.x_4$":
  - Each x_i is a decimal integer in the range [0, 255]
  - x_i cannot contain leading zeros
  - Examples: 192.168.0.1 (valid), 192.168.01.1 (invalid), 192.168@1.1 (invalid)

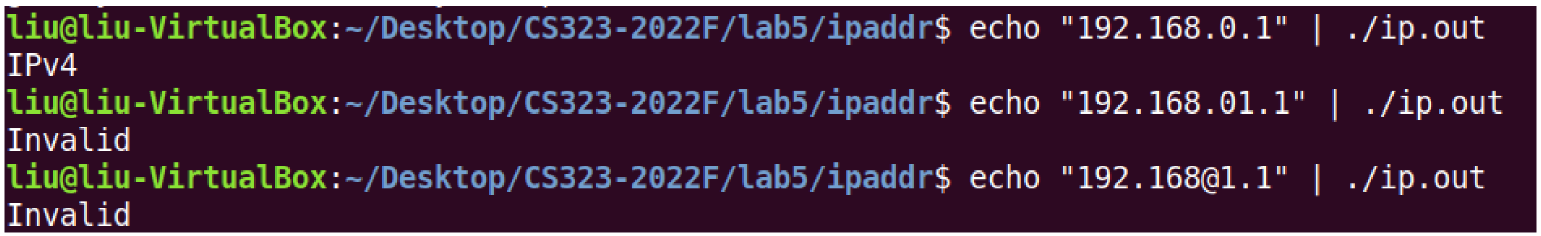
Validate IP Address (leetcode #468)

- A valid IPv6 address is an IP in the form "x_1:x_2:x_3:x_4:x_5:x_6:x_7:x_8":
  - The length of each x_i is in the range [1, 4]
  - x_i is a hexadecimal string which may contain digits, lowercase English letter ('a' to 'f') and upper-case English letters ('A' to 'F')
  - Valid examples:
    - `2001:0db8:85a3:0000:0000:8a2e:0370:7334`
    - `2001:db8:85a3:0:0:8A2E:0370:7334`
  - Invalid examples:
    - `2001:0db8:85a3::8A2E:037j:7334`
    - `02001:0db8:85a3:0000:0000:8a2e:0370:7334`

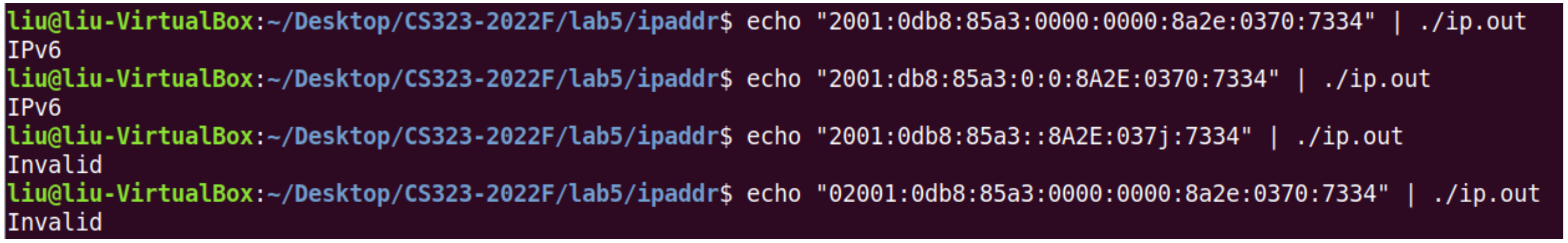
More instructions

- Clone the `lab7/ipaddr` directory
- The `lex.l` file is provided to recognize x strings (but does not check its validity), the dot and colon in IP addresses. Please use it as is.
- Complete the `syntax.y` file and providing production rules, semantic actions, as well as necessary supporting functions
- You may use the build target `ip` to get the executable `ip.out`

Test Inputs and Sample Outputs



```
liu@liu-VirtualBox:~/Desktop/CS323-2022F/lab5/ipaddr$ echo "192.168.0.1" | ./ip.out
IPv4
liu@liu-VirtualBox:~/Desktop/CS323-2022F/lab5/ipaddr$ echo "192.168.01.1" | ./ip.out
Invalid
liu@liu-VirtualBox:~/Desktop/CS323-2022F/lab5/ipaddr$ echo "192.168@1.1" | ./ip.out
Invalid
```



```
liu@liu-VirtualBox:~/Desktop/CS323-2022F/lab5/ipaddr$ echo "2001:0db8:85a3:0000:0000:8a2e:0370:7334" | ./ip.out
IPv6
liu@liu-VirtualBox:~/Desktop/CS323-2022F/lab5/ipaddr$ echo "2001:db8:85a3:0:0:8A2E:0370:7334" | ./ip.out
IPv6
liu@liu-VirtualBox:~/Desktop/CS323-2022F/lab5/ipaddr$ echo "2001:0db8:85a3::8A2E:037j:7334" | ./ip.out
Invalid
liu@liu-VirtualBox:~/Desktop/CS323-2022F/lab5/ipaddr$ echo "02001:0db8:85a3:0000:0000:8a2e:0370:7334" | ./ip.out
Invalid
```

Project Reminder #2

- Please start to design and implement your language if you haven't done so.
  - It is also fine if you choose to build a compiler for SPL or its simple variants.

- Milestone check: Nov. 18, during the lab session.
  - Please prepare test cases by yourself for the demo.
  - You should also prepare a report, which should at least contain: 1) the specification and core features of your language, 2) the design of your compiler, 3) the implementation progress.