COEN 175

Phase 2 - Week 1

TA

- Antonio Gigliotti: agigliotti@scu.edu
  - Office Hours: Thursday 11 - 1 PM

Extra Help/Tutoring

- Tau Beta Pi Tutoring
  - Wednesday 2:30 - 3:30 PM
- Link to Tutoring schedule and zoom link
  - https://sites.google.com/scu.edu/scutaubetapi/tutoring?authuser=1&pli=1

Phase 2 - Syntax Analysis

1. Disambiguate expression Grammar
2. Modify lexer.l to return tokens from tokens.h
3. Test new lexer with lextest.cpp
4. Write parser.cpp for expressions

- Due 11:59PM on Sunday, April 18th
- Completing half this week

1. Disambiguate Expression Grammar

- Use the operator associativity/precedence table to disambiguate all of the expression grammar on the phase 2 assignment

$$\begin{aligned} E &\longrightarrow E * E \\ &\mid E + E \\ &\mid E - E \\ &\mid E \,/\, E \\ &\mid \mathbf{id} \end{aligned} \quad \xrightarrow{\textit{Disambiguate}} \quad \begin{aligned} E &\longrightarrow E + T \\ &\mid E - T \\ &\mid T \\ T &\longrightarrow T * F \\ &\mid T \,/\, F \\ &\mid F \\ F &\longrightarrow \mathbf{id} \end{aligned} \quad \xrightarrow[\textit{Recursion}]{\textit{Eliminating Left}} \quad \begin{aligned} E &\longrightarrow T\,E' \\ E' &\longrightarrow +\,T\,E' \\ &\mid -\,T\,E' \\ &\mid \varepsilon \\ T &\longrightarrow F\,T' \\ T' &\longrightarrow *\,F\,T' \\ &\mid /\,F\,T' \\ &\mid \varepsilon \\ F &\longrightarrow \mathbf{id} \end{aligned}$$

2. Modify lexer.l

- Start from phase 1 solutions
  - Download solution.tar from camino (Project → 1)
- Edit **tokens.h** to include all tokens
  - All unique operators (e.g. +,-,/,%)
  - ID, num, string, done, error
  - All keywords (given)
- Modify **lexer.l**
  - Return appropriate token instead of printing them out
    - Ex. *return AUTO* instead of calling *printToken("keyword")*
  - Single char operators
    - *return *yytext*

3. Test New Lexer

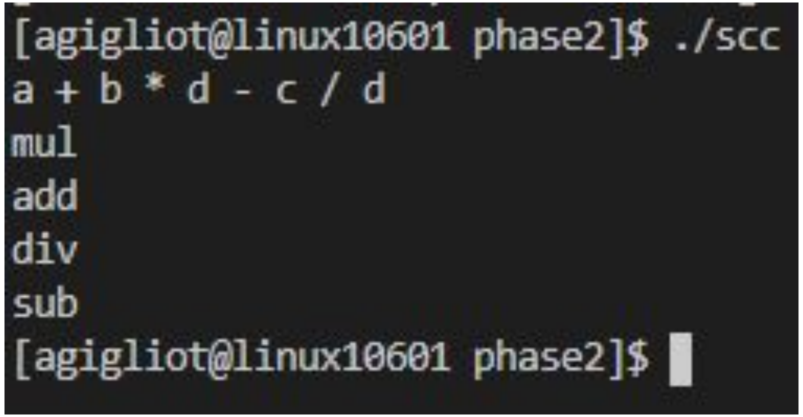
- Download **lextest.cpp** from camino (labs → 2)
- Modify Makefile to compile with *lextest* instead of *parser*
  - Replace *parser.o* with *lextest.o*
- Run phase 1 examples with the new lexer and lextest to confirm that your new lexer is ready to go
  - Should not need to edit lexer again after this point

- Once you confirm that your lexer is working, no need to use lextest.cpp anymore (replace *lextest.o* with *parser.o* in Makefile)

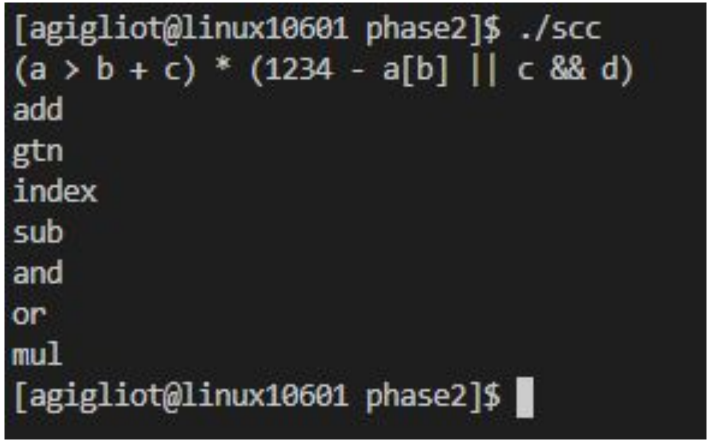
4. Writing the Parser

- Remember to import **lexer.h** and **tokens.h**
- Write your main() and match() functions (need to declare a global **int lookahead**)
  - Read lecture 4 slides for examples
- Write the code for expressions:
  - Start with algebraic binary (+, -, *, /, %)
  - Then prefix (!, &, …)
  - The rest of expressions
- Remember to print out the output for each operator once the whole operation has been matched and completed (assignment doc has the required output for each operator)
- Goal is to have expression written by beginning of next lab section (can hold off on *cast* and *sizeof* until next week if you'd like)

Examples



```
[agigliot@linux10601 phase2]$ ./scc
a + b * d - c / d
mul
add
div
sub
[agigliot@linux10601 phase2]$
```



```
[agigliot@linux10601 phase2]$ ./scc
(a > b + c) * (1234 - a[b] || c && d)
add
gtn
index
sub
and
or
mul
[agigliot@linux10601 phase2]$
```

- **ctrl+d** to end input