

Project Part #1: The Scanner

Due: Thursday, February 15

Now it is the time to start applying all the neat tricks we learned in Chapter 2 of the text to implement the scanner. First, make a new directory called **Scanner** under your CS440 directory to place your work.

1. You will need to copy 6 of the TINY files plus Makefile into the new directory and modify them to be used as template for the Scanner.
2. In order to have consistent output, edit the **main.c** file to set **TraceScan** to **TRUE**, **EchoSource** to **TRUE** and **NO_PARSE** to **TRUE**. Before you move on, it makes sense to recompile and run the TINY compiler on **sample.tny**. Verify that the tokens are written to the screen, and that no code is generated.
3. Design your scanner as a DFA on paper.
4. Decide whether you want to implement the C- scanner using the technique used in TINY (the **nested case**-statements like in Figure 2.7 page 62) or the **table-driven** technique (described in page 63).
5. Decide whether you want to implement the lookup function using binary search or using a hash function.
6. Implement and test your code thoroughly. Write sample C- code to test your scanner on. Additional sample programs are on p. 496 of the book. For testing the scanner, the C- programs do not need to be syntactically or semantically correct. Demonstrate that you test the generation of every token and every error message you should produce. Remember that the C- source code file should have the **.cm** extension.
7. Make sure you have appropriate comments in your files including header comments describing the name(s) of the file creator/modifier, date of modifications and summary of the file contents and use. When needed, have in-line comments to explain elaborate coding that your instructor or your colleagues might have troubles understand.
8. Hand in the DFA, printouts of your code and your tests.