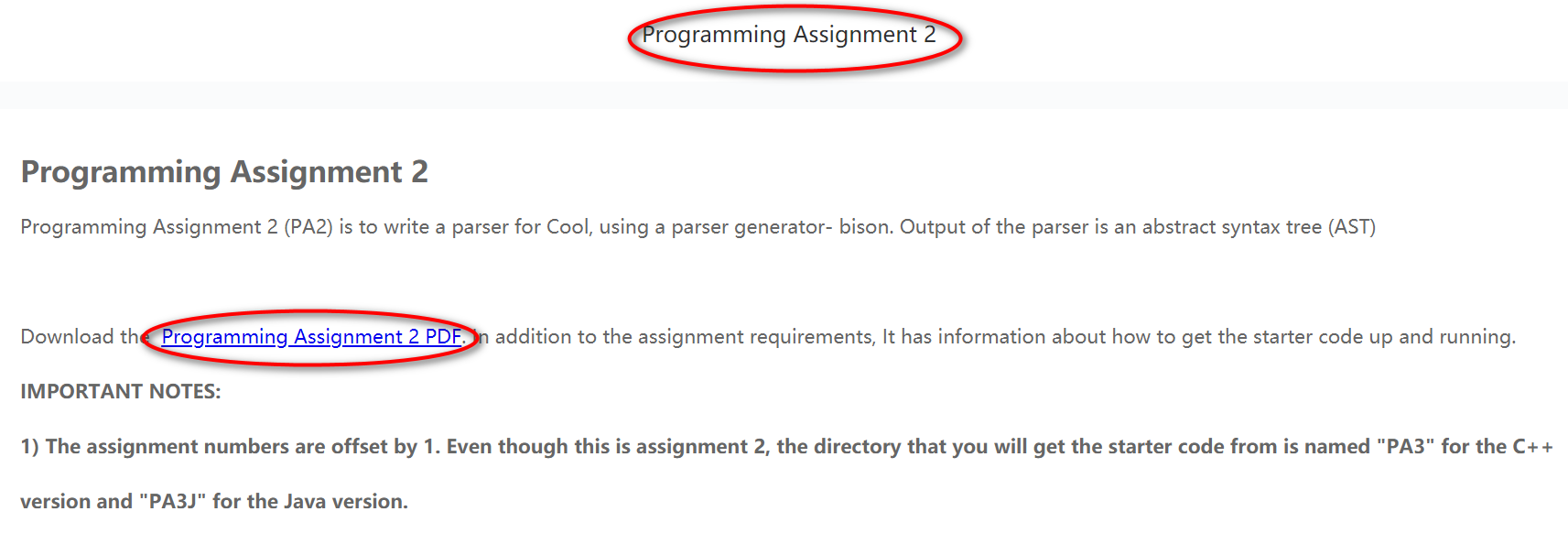
How to do Programming Assignment2

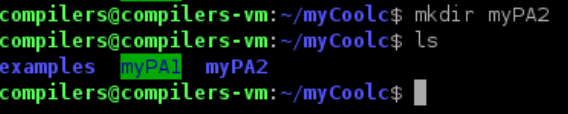
Programming Assignment 2 (PA2) is to write a parser for Cool, using a parser generator- bison. Output of the parser is an abstract syntax tree (AST)

How to do:

1. Go to course website and read the instruction of PA2. Download the Assignment 2 PDF.



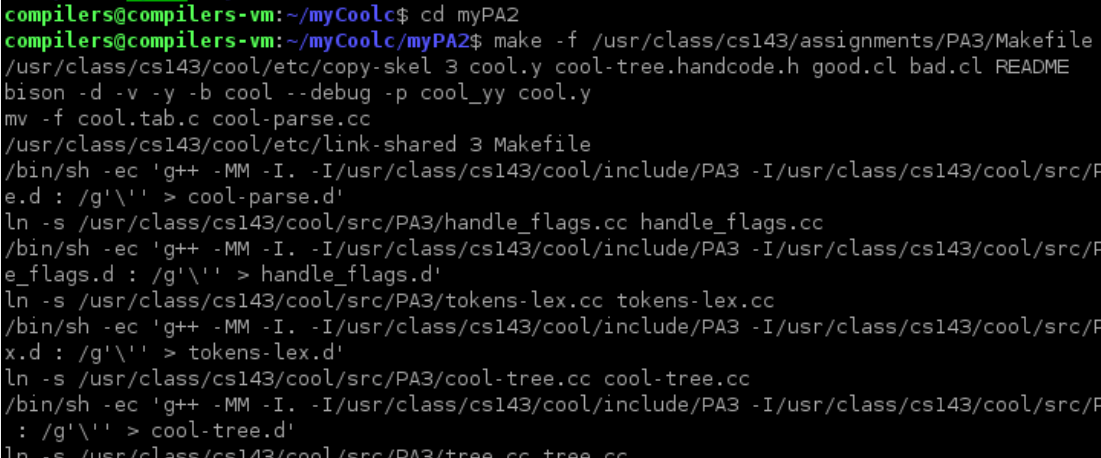
1. To get started, create a directory where you want to do this assignment. For example, create a directory named “myPA2”



In “myPA2” directory, execute the following command to get the starter code

“make -f /usr/class/cs143/assignments/PA3/Makefile”

Note: The assignment numbers are offset by 1. Even though this is assignment 2, the directory that you will get the starter code from is named "PA3"



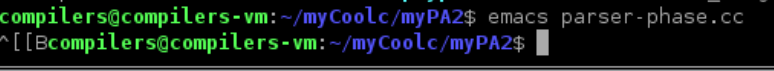
This command will copy a number of files to your directory as the starter code.

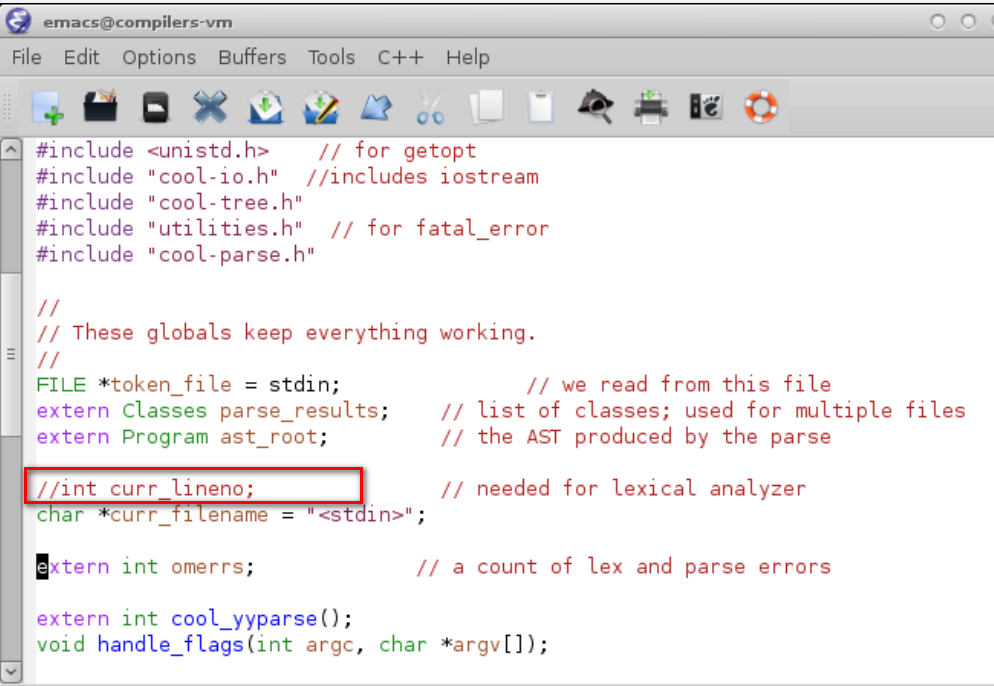


**Important**: you will need to make one slight change to our code before it will link. Please comment out line 29 of the file parser-phase.cc (which you should not otherwise modify), so that it looks like:

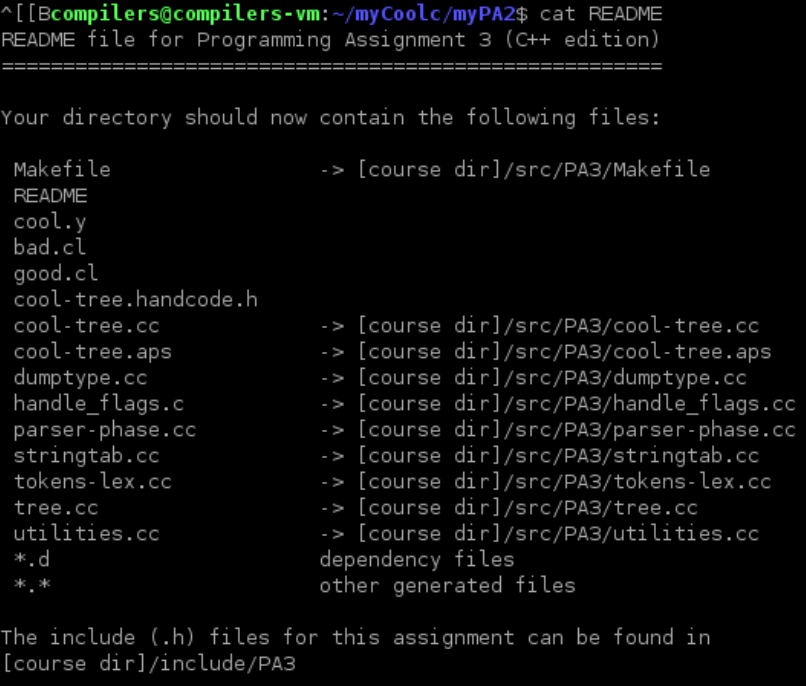
//int curr\_lineno; // needed for lexical analyzer

Using command “emacs parser-phase.cc”

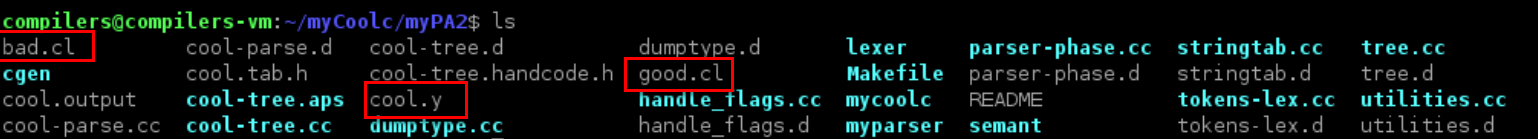




1. Read the README file. This file contains detailed instructions for the assignment as well as a number of useful tips.



1. What you need to do is to modify file “cool.y”, “good.cl” and “bad.cl”



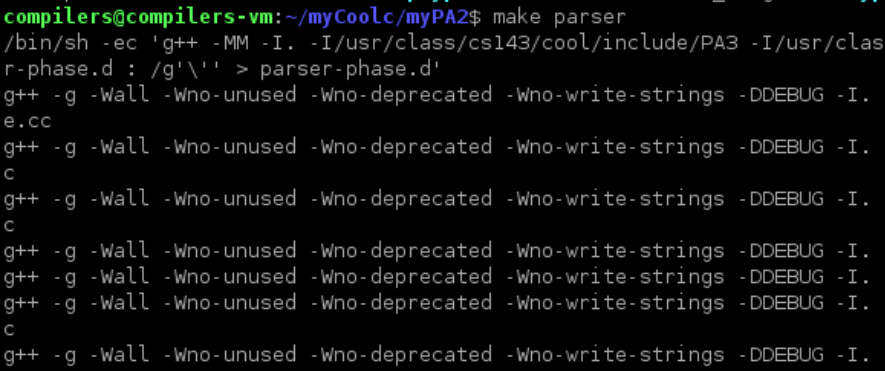
cool.y is the skeleton for the parser specification that you are to write. It already contains productions for the program and the classes. Use them as an example to write the remaining productions.

good.cl and bad.cl are used to test a few features of the grammar. You should add tests to ensure that good.cl exercises every legal construction of the grammar and bad.cl exercises as many different parsing errors as you can squeeze into one file.

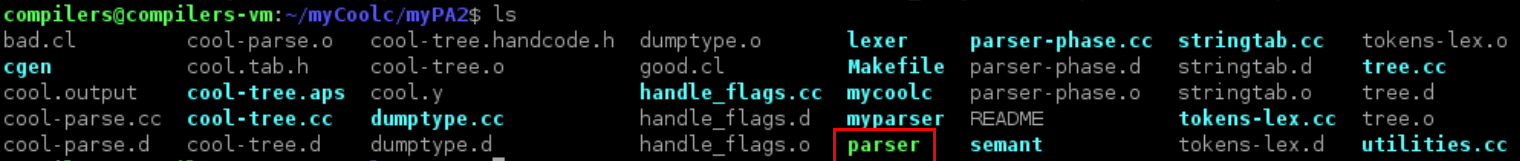
Before you do any modification, although these files are incomplete as given, the parser does compile and run.

1. Build the parser and run it.

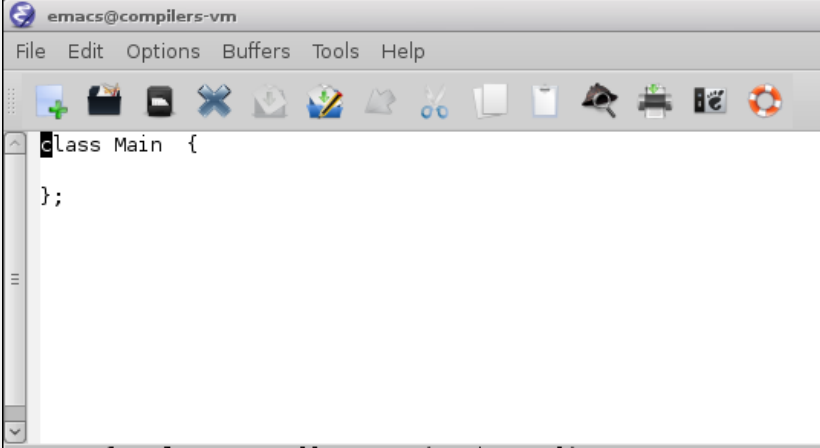
To compile the parser, type command “make parser”



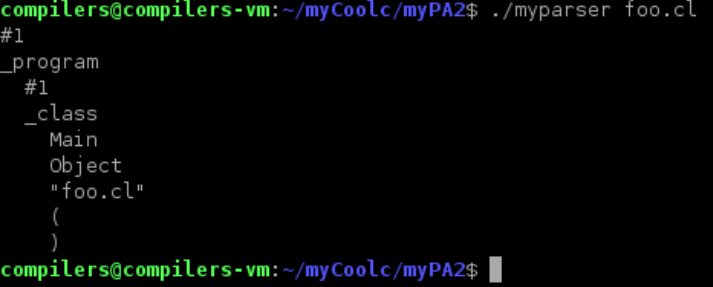
This produces an executable file named "parser" which is standalone phase of the Cool compiler. It requires lexer, semant, and cgen to do anything useful.



When running, “parser” needs an input, that is a “.cl” file. Let’s create a new file “foo.cl” using command “emacs foo.cl”. Write some simple code in it.

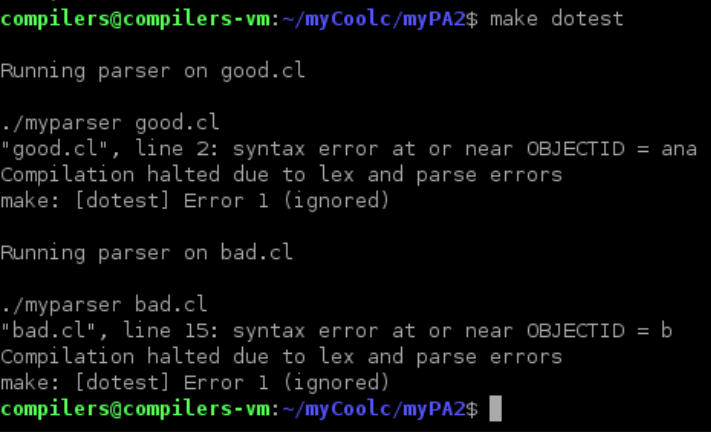


To test your parser on the file 'foo.cl' type “./myparser foo.cl”. “myparser” is a shell script that "glues" together lexer and parser using pipes.



You can try command “./myparser -p foo.cl”. Flag “p” for debugging the parser; using this flag causes lots of information about what the parser is doing to be printed on stdout.

To run your parser on the files good.cl and bad.cl type “make dotest”

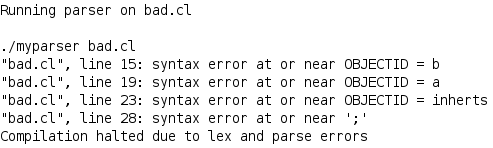
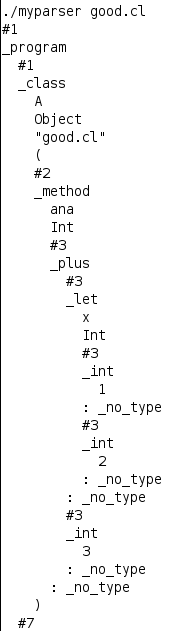


Since we haven’t do anything in file “cool.y”, these two files cannot be parsed properly.

1. The Parser Output

* For programs that parse successfully, the output is a listing of the AST
* For programs that have errors, the output is the error messages.

When you are done with cool.y, the outputs of parsing good.cl and bad.cl look like:



For building Abstract Syntax Tree (AST) of Cool, you need to consult “A Tour of Cool Support Code.pdf” Session 6 Abstract Syntax Trees

1. Where to start:

You can follow the following steps:

In the first week, do:

* Step1: Add type declarations for non-terminals, add precedence declarations. To get the precedence of operations, read file “The Cool Reference Manual.pdf” Session “11.1 Precedence”
* Step2: Turn the cool syntax to yacc rules, leaving alone the code for AST generation.

In the second week, do:

* Step3: Error handling

In the third week, do:

* Seep4: Add the actions for AST generation to rules （**this part is optional**）

Let me give an example for parsing “if-else condition”

* Step1: Add type declaration **%type <expression> expression** if it hasn’t been added in the type declaration part.
* Step2: Add the rules

**expression : IF expression THEN expression ELSE expression FI**

* Step3: Error handling

**expression : IF expression THEN expression ELSE expression FI**

**| IF error**

**| IF expression THEN error**

**| IF expression THEN expression ELSE error**

* Step4: Add the code for AST generation

**expression : IF expression THEN expression ELSE expression FI**

**{ $$= cond($2, $4, $6);}**

**| IF error**

**| IF expression THEN error**

**| IF expression THEN expression ELSE error**

You can put AST and error handling aside first in the first week. Just add rules according to cool grammar syntax.

1. Grading your parser

Download the grading script and put it in the directory where the cool.y file is.

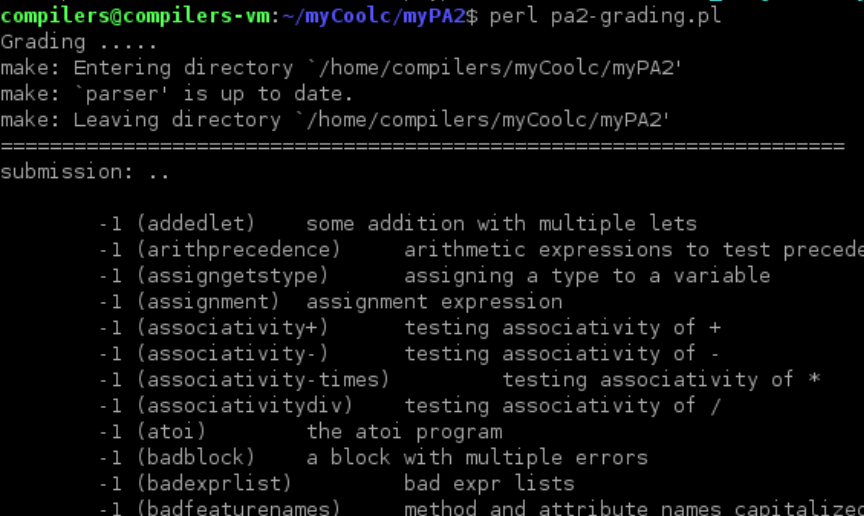
Using the following command to download:

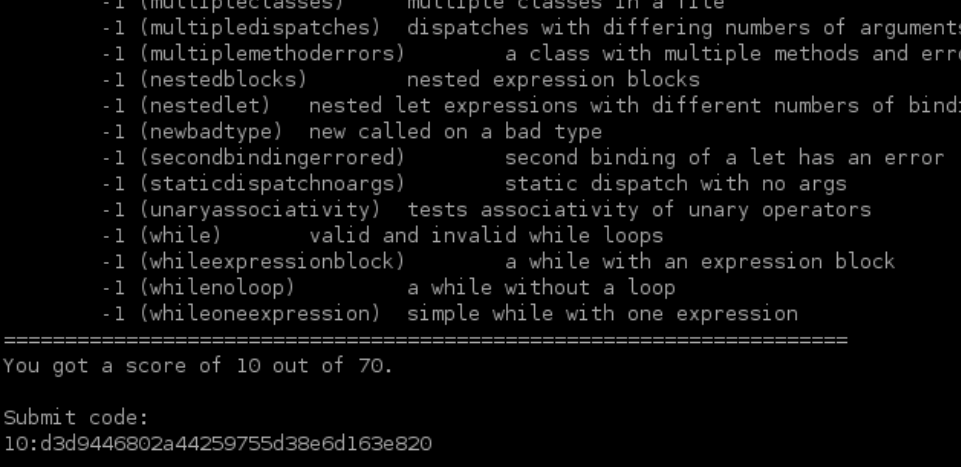
wget https://courses.edx.org/asset-v1:StanfordOnline+SOE.YCSCS1+1T2020+type@asset+block/pa2-grading.pl

This will save the script (pa2-grading.pl) in your assignment directory.



Run the script by typing “perl pa2-grading.pl”. The script will give you a grade at the end.





If you want to figure out why your parser is failing certain tests, the tests will be put in the ./grading subdirectory.

The output from your code will be in the ./grading/test-output directory.

Once you are satisfied with your grade, you can submit your result.

1. Submit your result.

* You are required to submit your cool.y file, together with your experiment report.
* Name your cool.y as学号+姓名+ cool.y
* Name your report as 学号+姓名+Experiment2.docx
* When preparing your experiment report, follow the “Experiment Report Template.docx”