COMS W4115
Programming Languages and Translators

Spring 2018

https://courseworks2.columbia.edu/courses/53311

Class Sessions	Mondays and Wednesdays, 2:40pm-3:55pm, Mudd 833
Instructor	Richard Townsend, rtownsend@cs.columbia.edu, CSB 467

1 Course Overview

Summary

In COMS 4115, you will learn about both the structure of computer programming languages and the basics of implementing compilers for such languages. Features will be covered from various programming paradigms, including imperative, object-oriented, functional, and concurrent languages.

The focus of the course is an intensive, semester-long project: you and a team will design a small programming language and implement a compiler for it using the OCaml language. The algorithms and concepts you will learn have broad application outside of the course: many programming tasks can be understood as variations of interpretation or translation, and understanding how a compiler operates will further develop your abstract thinking skills and make you a better programmer.

Prerequisites

- COMS W3157 Advanced Programming: You will be dividing into teams to build a compiler, so you need to have some idea how to keep this under control (e.g., Makefiles and source code control systems).
- COMS W3261 Computer Science Theory: You will need a formal understanding of regular expressions and context-free grammars to build the front-end of your compiler.

Course Goals

By the end of this course, students should be able to

1. appraise the benefits and limitations of functional programming.
2. relate formal language theory to front-end compiler implementation.
3. describe how a compiler implements different language features (first-class functions, objects, user-defined types) and designs (type systems, evaluation strategies, memory management systems).
4. motivate and describe the four essential phases of a compiler (scanning, parsing, semantic checking, code generation).
5. recognize the difficulties faced by a group working on a large, long-term software project, and develop potential solutions to those difficulties.

2 Technical Resources

Office Hours

Name	Time	Location
Richard Townsend	Fri. 10:15am-12:15pm	CSB 469
Jennifer Bi	Tues. 4:00pm-6:00pm	CSB 468
Wode "Nimo" Ni	Wed. 4:30pm-6:30pm	CSB 468
Rabia Akhtar	Thurs. 5:00pm-7:00pm	CSB 468
Chang Liu	Mon. 10:00am-12:00pm	CS TA Room
Jordan Vega	Wed. 11:00am-1:00pm	CS TA Room

The full Office Hours schedule is posted and kept up to date on the course calendar.

Piazza

We will be using Piazza for class discussion; a link to the course Piazza page can be found on CourseWorks. Please post all technical questions on Piazza (instead of emailing me or a TA); post publicly if at all possible, as other students may have similar questions or be able to help you faster than the TAs or instructor. Please post privately to Instructors if your question reveals individual work.

Suggested Textbook(s)

While this course has no required textbook, the following references can help facilitate your understanding of the covered content.

- *Compilers: Principles, Techniques, and Tools, 2nd Edition.* Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Addison-Wesley, 2006.

  The "dragon book" has long been touted as the definitive textbook on the theory of compiler design. While some sections have become dated due to language evolution, many are still completely applicable today (especially those on front-end compiler design and compiler optimizations). Columbia's own Prof. Al Aho is one of the authors!

- *Programming Language Pragmatics, 4th Edition.* Michael L. Scott. Morgan Kaufmann, 2015.

  A broad-minded book about languages in general, but has less on practical details of compiler design.

- *Modern Compiler Implementation in Java, 2nd Edition.* Andrew W. Appel. Cambridge University Press, 2002.

  The opposite of Scott; this book focuses on how to actually implement a compiler in Java (Appel also published ML- and C-based textbooks).

Lecture Materials
All slides will be posted on CourseWorks within a day of their presentation in class. They can be found in the "Files" section of CourseWorks, with links to appropriate slidedecks in the "Modules" section.

We will be using the PollEverywhere software during class sessions to facilitate an active learning environment and give the instructor a sense of class-wide comprehension. After the add/drop period ends, each student will be given a PollEverywhere account to be linked with their CourseWorks account for the class. We will cover the use of PollEverywhere in our first class session.

OCaml resources
Start at the OCaml homepage to find compiler installation instructions and documentation and the user's manual for the whole system, including ocamllex, ocamlyacc, and all standard libraries.

OCaml tutorials: Real World OCaml is a cogent, well-written textbook (completely online) that covers the OCaml language; the first chapter is an excellent complement to my OCaml slides. The official website also has a collection of tutorials, but these are more loosely organized.

Project resources
Stephen Edwards has been teaching this class for years; check out his previous class webpages, where he lists past groups' project deliverables and compiler files.

The success of your project is strongly tied to your ability to use version control systems like git and github. These tools will let you collaborate on code files, track each member's contributions and modifications to the project, and ensure everyone has access to the same codebase. Here are some tutorials:

- An intro to git and github for beginners.
- A collection of guides for github use.
- Full documentation on the git version control system.

We will not be covering git in class; please post on Piazza or attend Office Hours if you need assistance setting up or using a version control system.

3 Tentative Schedule

Week	Date	Topic	Assignments	
			HW	Project
1	Jan. 17	Course Overview and Intro. to Languages		
2	Jan. 22	Language Processors		
	Jan. 24	Essential OCaml I		
3	Jan. 29	Essential OCaml II		
	Jan. 31	Essential OCaml III		
	Feb. 2	(Proposal due only)		Proposal
4	Feb. 5	Syntax, Lexing, and Parsing I		
	Feb. 7	Syntax, Lexing, and Parsing II	HW 1	
5	Feb. 12	Syntax, Lexing, and Parsing III		
	Feb. 14	QUIZ 1		
6	Feb. 19	Syntax, Lexing, and Parsing IV		
	Feb. 21	Parsing V; Semantic Analysis I		Scanner & Parser
7	Feb. 26	Semantic Analysis II		LRM
	Feb. 28	Semantic Analysis III; Code Generation and LLVM I	HW 2	
8	Mar. 5	Code Generation and LLVM II		
	Mar. 7	QUIZ 2		
9	Mar. 12 Mar. 14	Spring Recess		
10	Mar. 19	Code Generation and LLVM III; Mid-Course Feedback		
	Mar. 21	Snow Day		
11	Mar. 26	Code Generation and LLVM IV		Hello World
	Mar. 28	Codegen V; Types and Type Systems I	HW 3	
12	Apr. 2	Types and Type Systems II		
	Apr. 4	QUIZ 3		
13	Apr. 9	Runtime Environments I		
	Apr. 11	Runtime Environments II		
14	Apr. 16	Runtime Environments III		
	Apr. 18	Naming Issues (if we have time)		Extended Testsuite
15	Apr. 23	The Lambda Calculus I	HW 4	
	Apr. 25	The Lambda Calculus II		
16	Apr. 30	QUIZ 4		
	May 7-9	Presentations		
	May 9	Project Report Due		

The "Buffer" classes listed on April 23 and 25 are meant to give the course schedule some flexibility; if more time is required for some earlier concepts, the schedule can be pushed back without sacrificing later content. If the course ends up moving faster or as anticipated, the content of these two class sessions will be determined jointly by the instructor and students.

Update, 2/14/18: We'll be using Feb. 19's session to finish our Parsing slides, so all class session topics have been pushed forward by one session and one Buffer Class has been removed.

Update, 3/14/18: The schedule after Spring Recess has been overhauled; the class topics have been modified, and the 5th compiler deliverable has been changed and moved to a later date.

Update, 3/19/18: The schedule has been updated to deal with Columbia's snow day.

4 Assessments and Grading

Assessment will be based on five criteria:

1. One semester-long group project (40%): Design the goals, syntax, and semantics of a little language, and implement a compiler for that language. Deliverables are due on Feb. 2, Feb. 21, Feb. 26, Mar. 26, Apr. 18, and May 9.
2. Four in-class quizzes (25%): These will occur during the last 40 minutes of class on Feb. 14, Mar. 7, Apr. 4, and Apr. 30.
3. Four Homework Assignments (15%): These written assignments will gauge your understanding of theoretical compiler concepts and further prepare you for the project. Assignments are due on Feb. 7, Feb. 28, Mar. 28, and Apr. 23.
4. Participation (10%): We will be using the PollEverywhere software to facilitate interactive learning in most class sessions. You are expected to attend class sessions to answer the questions posed.
5. Peer-Assessment (10%): As part of your final project deliverable, you will privately assess your teammates' contributions to the project as a whole. Your teammates' assessments will factor into your final grade.

Homework and Project Deliverable Logistics

All homeworks and project deliverables must be submitted via CourseWorks. No submissions will be accepted via email or other means. Project deliverables will be submitted as a group, i.e., only one member will submit the deliverable.

Homeworks and project deliverables are due at 11:59:00PM on their due date. To avoid missing this strict deadline, we advise submitting whatever you have a bit earlier even if incomplete; you may submit as many times as you wish before the deadline. We will always grade the latest submission.

Each homework assignment and deliverable will have specific instructions posted on CourseWorks. These instructions must be followed exactly to receive a full grade.

Late Policy

All students have a budget of three free late days for any deliverable (homework or project) *except the proposal and the final project report*. You may only use at most two late days on any given deliverable, after which you'll be given a zero for that assignment.

Each late day is exactly 24 hours: if your submission is timestamped anywhere between 11:59:01PM on the due date to 11:58:59PM the following day, that counts as the use of a late day. If a group deliverable is late, that will use up a late day *for each group member*.

After your three free late days are used up, you may still submit assignments late with a 15 point penalty per day. Again, submissions received after two late days will receive a zero.

Regrades

If you want to request a regrade for an objective error on our part, you must submit a private request to the Instructors via Piazza explaining the error. The deadline for these requests is one week from the time the grades were posted for a given deliverable; no exceptions to this policy will be made.

5 Collaboration and Academic Honesty

If you have **any** questions about the below policies or a specific situation dealing with academic integrity, do not hesitate to ask the instructor. All students are expected to read and adhere to the Computer Science Department's Academic Honesty Policy.

Collaboration: Projects

You will collaborate with a small team on the semester-long project; your approaches and designs may be shared with other teams, but no code, text, or images should be shared, partially or otherwise.

Collaboration: Non-Project Course Work

All homework assignments and in-class quizzes are *individual work*: while you may discuss the problems and general strategies with classmates, each student is expected to ultimately come up with their own solution. Lifting partial or complete solutions from anyone (classmates, online sources, strangers) is completely prohibited. To ensure you follow this policy, nothing should be written down, typed, or recorded in any way from your discussions with other students about homeworks. You may not work on the specific problems with anyone other than TAs or the instructor, and you should not use the internet in any capacity to help solve any problem. No questions, student solutions, or instructor-provided solutions should be posted online in any capacity (except as a submission to CourseWorks).

Academic Honesty

Any violation of the above policies will be considered cheating, and all students involved in any capacity will be referred to both the Computer Science Academic Committee as well as to the relevant Dean.