CIS 624: Structure of Programming Languages
Fall 2020

Logistics and Contact Information: The instructor is Boyana Norris. See the course Canvas page for all pertinent logistics information and up-to-date course materials, including lecture recordings.

Instructor: Boyana Norris; Time: 8:15-9:45 MW

Office hours: TBD

Learning objectives: We will investigate concepts essential to programming languages including control flow, scope, functions, types, objects, and threads. As time permits, we will explore other concepts such as continuations, exceptions, message passing, and connections to logic. Our primary intellectual tools will be operational semantics and OCaml programs. Prior knowledge of neither is required nor expected. We will apply the basic concepts learned in lecture and small homework projects to a problem involving a real domain-specific language (DSL) design and implementation, culminating in the formal definition of its semantics definition and interpreter implementation.

Successful course participants will learn to:

- Give precise definitions to programming-language features that can be used to prove properties of inductively defined sets (e.g., well-typed programs).
- Define and implement domain-specific languages.
- Appreciate some programming-language theory jargon (e.g., inference rules) and find the research literature more approachable.
- Write better programs by exploiting modern language features such as higher-order functions and objects.

In short, our goal is to use theory to make us better programmers and better researchers. Always think, "how is this related to programs I have written?"

FORMAT

Two weekly lectures will develop the course content. The recommended textbooks (Types and Programming Languages by Pierce) covers some of the same material. References to other (free) materials will be added throughout the term. It can serve as an excellent second explanation, but we may not necessarily follow it closely, so you can consider it optional. Homeworks will extend the material discussed in lectures; expect to learn as you do them. Programming exercises must be done in the OCaml language, which will be discussed in class.

Grading:

- There will be 5-7 incremental weekly projects, which will contribute a total of 60% to the course grade.
- 10% of the course grade will be based on participation in online lecture and asynchronous discussion activities; grading will be done based on your attendance and contribution to in-lecture group mini-projects.
• The final group project will contain both theoretical and software components and will be worth 30% of the course grade.
• There will be no exams in this course.

**Academic Integrity:** Any attempt to misrepresent the work that you have done will result in failing this course. If there is any doubt, ask the instructor in advance and make sure to indicate on your assignment who assisted you and how. In general, you may discuss general approaches to solutions, but unless the assignment explicitly states to work in pairs or groups, **you must write your solutions on your own.** You should not show your written solution to someone else or view someone else's solution. Particular assignments may include more specific instructions. If not, do ask. Violating the academic trust your instructor and classmates have placed in you will have a **far worse** effect on your academic future than doing poorly on a homework assignment.

**Accessible Education.** The University of Oregon is working to create inclusive learning environments. Please notify me if there are aspects of the instruction or design of this course that result in disability-related barriers to your participation. You are also encouraged to contact the Accessible Education Center in 164 Oregon Hall at 541-346-1155 or uoaec@uoregon.edu.

**Course materials**

Required: None

Recommended:

• Programming Language Foundations, Aaron Stump, 2013, Wiley (selected portions available in PDF form on Canvas, so you are not required to obtain this book)
• **DSL Engineering**, by Markus Voelter, also online.

Suggested:

• *Programming Languages : Theory and Practice*, Robert Harper, draft available online, also on Canvas
• *Syntax and Semantics of Programming Languages*, Kent Petersson
• Types and Programming Languages, Benjamin C. Pierce, 2002, MIT Press. [Errata](#)
• The Formal Semantics of Programming Languages, Glynn Winskel, 1993, MIT Press

Additional references will be offered throughout the course.

**TOPICS**

• Introduction to OCaml
• Inductive definitions
• Operational semantics
• "Pseudodenotational" semantics
• Assignment and basic control flow
• Scope and variable binding
• Continuation passing style and CPS transform
• Simple types
• Type safety
• Domain-specific languages
• Interpreters