

CS 421 – Programming Languages and Compilers

Welcome!

Today's class:

- ▶ Staff
- ▶ Why 421?
- ▶ Class structure and policies
- ▶ This week's assignments
- ▶ Overview of languages



Staff

Professor: Sam Kamin, 4237 SC, 3-8069
Office hrs: TBD

TAs: Chris Osborn (TBD)
William Mansky (TBD)

Web page:

<http://www.cs.illinois.edu/class/cs421/>

Newsgroup: news.cs.uiuc.edu – class.cs421



What is 421 about?

Implementation and design of programming languages

First half of course: Implementation

- ▶ Languages implemented by programs called *compilers*, together with *run-time systems* that provide necessary services at run time. *We will study compilers and run-time systems.*
- ▶ Compilers have to *parse* programs, and then *translate* them to an executable form. *We will study parsing and translation.*

Second half of course: Design

- ▶ There are many languages in use today. *We will study a variety of languages.*
- ▶ Languages are complicated, but can be understood by formal definition and analysis. *We will study formal definitions of languages, especially their type systems and operational behavior.*



Why?

Why learn about compilers?

- ▶ Complete picture of how programs go from keyboard to execution
- ▶ Understand translation from high-level language to machine language
- ▶ Learn to build compilers and other programs that process structured input
- ▶ Learn interesting algorithms



Why?

Why learn about languages?

- ▶ Increase ability to learn new languages
 - ▶ Learn correct terminology for describing languages
 - ▶ Become better programmers by seeing different perspectives on programming
-
- ▶

Class structure

- ▶ Lectures Tuesday and Thursday. Plain notes (usually) posted before class; annotated notes posted after class. *Strongly advise taking notes in class.*
 - ▶ Weekly assignments
 - ▶ Programming in OCaml
 - ▶ Grades
 - ▶ Cheating
 - ▶ Lateness
-
- ▶

Class structure

Topics:

- ▶ **First half: Compilers**
 - ▶ Next 2 classes: OCaml
 - ▶ Then: Lexical analysis and parsing (applicable also to non-PLs)
 - ▶ Then: Creating executable code
- ▶ **Second half: Languages**



OCaml

We will use the OCaml programming language for most of the work in this class

- ▶ **Functional programming language**
 - ▶ One of the two leading language paradigms (the other is object-oriented)
 - ▶ Defined mainly by *no assignment statements*
 - ▶ Heavy use of *dynamically-allocated data structures* and *recursion*
- ▶ **Everything we will do in the first half of the class could also be done in Java, but:**
 - ▶ OCaml notationally more concise
 - ▶ Using OCaml now will prepare you for more advanced uses of OCaml in second half of class



This week's assignments

- ▶ **MP0.**
 - ▶ Handout on basic OCaml.
 - ▶ Simple programming assignment.
 - ▶ “Due” Wednesday night. This assignment is not graded, but we have set up the handin program for it so that you can get used to it.
 - ▶ **MPI.**
 - ▶ Based on material we will cover on Thursday.
 - ▶ Due before next Tuesday's (1/26) class.
-

Programming: Where were we, and where are we?

- ▶ **Where we were**
 - ▶ Small, slow, serial computers
 - ▶ Little infrastructure beyond raw machine
 - ▶ Parallelism the exception rather than the rule
 - ▶ Emphasis on machine, rather than programmer, efficiency
 - ▶ Programmer control over memory usage
 - ▶ **Where we are**
 - ▶ Large, fast, multicore computers
 - ▶ Vast layers of functionality to build on
 - ▶ Emphasis on programmer efficiency, rapid development
 - ▶ Automatic memory mgt; built-in concurrency/parallelism
 - ▶ **Actually, now we're in both places at once: need high efficiency sometimes, rapid development other times –**
 - ▶ **and really want to have both all the time.**
-

What languages to people use today?

C, Java, C++, Objective C, C#, Python, Ruby,
Fortran, Javascript, OCaml, ...

- ▶ What distinguishes them from one another?



Capsule history of PLs

1957	Fortran		Lisp
	Algol		
1970	C		ML
1980	C++		OCaml
1990		Java	
			Python
2000		C#	Ruby
Now			Scala, Clojure, ...



Summary of PLs

	Traditional, “static”	Scripting, “dynamic”	Mixed
Examples	C, Fortran, C++	Python, Ruby, JS	Java, C#, OCaml
Applications			
Objects?			
Memory mgt.			
Compiles to			
Run-time system/JIT			
Concurrency			
Types			
Tagged values			