Old Dominion Univ.
Dept. of Computer Science

CS 355: Syllabus

Prof. Steven J. Zeil

Tues. & Thurs., 9:30–10:45AM

A printable version of this syllabus is available here.

Copyright © 2001-2003 zeil@cs.odu.edu
Last Revision Date: August 6, 2003
1. Basic Course Information

1.1. Objectives:

This course investigates the languages that we use to write programs. The purpose of this course is *not* to provide you with programming proficiency in a half-dozen or more languages. Although we will look at and write in many languages, you may not learn any of them in enough detail to do “real programming”.

Instead, you will learn the elements that make up programming languages, the rationale behind language design choices, and the way in which different language elements interact. Upon completing this course, you should be able to

- Learn new languages, on your own. In fact, you should never need to take another course just to learn an unfamiliar language.
- Evaluate the suitability of a language to a particular project.
- Anticipate the changes that are likely to take place in programming languages over the next several years.
1.2. Required Text:


Various other required or optional readings will be made available via the course web pages.

1.3. Prerequisites:

CS 250 (Problem Solving and Programming) or equivalent, CS 252 (Introduction to Unix for Programmers)

Students are presumed to be familiar with basic programming techniques, including the use of pointers, functions and procedures, loops and recursion. Also assumed is facility with basic algebra and logic.

Assignments in this course will employ a wide variety of programming languages. All required languages will be available on the Dept.’s Unix system. Where possible, the course web pages may indicate sources of language compilers and interpreters for other systems. In practice, however, small differences in compilers, hardware, and operating systems often make programs behave differently on different systems. In all cases, student programs will be graded based upon their
performance on the CS Dept’s Unix system, and, if a student develops his or her code using a different system, it is the student’s responsibility to port their code to the CS Dept. Unix system and to test it there.

2. Communications

Steven J. Zeil 248D Education Building
(757) 683-4928 Fax: (757) 683-4900
e-mail: zeil@cs.odu.edu

2.1. Course Repository and Home Page:

Most on-line materials for this course can be found at http://www.cs.odu.edu/~zeil/cs355/.

Files related to assignments, however, will be available via anonymous ftp at ftp.cs.odu.edu in /pub/zeil/cs355. Students logged into a CS Dept. Unix machine will also be able to find these files at string zeil/cs355.
Section 2: Communications

2.2. Computer & E-mail Accounts

Students must have the following accounts to participate in this course:

- A login account for the CS Dept.’s Unix network
- A CS Dept e-mail account

Please note that accounts that you use to access CS Dept. machines and ODU (OCCS) machines are not the same. Instructions for obtaining CS Dept. accounts may be found at the CS Dept. home page, [http://www.cs.odu.edu](http://www.cs.odu.edu) by clicking on “Account Creation”.

I will use electronic mail to your CS Dept account for timely communication, especially of clarification/corrections/changes to homework or projects. Students should check their e-mail on a regular basis.

2.3. Office Hours:

Weekly office hours are posted on a calendar accessible from my [course web page](http://www.cs.odu.edu).
3. Assignments

Students should expect a short assignment every 1-2 weeks. Roughly half of these will be programming assignments; the rest will be written problem sets.

Students should anticipate that programming assignments will require them to learn details of programming language syntax and semantics beyond what is covered in the text. Appropriate reference manuals will be kept in the course repository for this purpose. Because working in a new language is always time-consuming, extra time will be allotted beyond what would normally be required if only the complexity of the program itself were taken into consideration. This may well mean that programming and written assignments will overlap.

4. Course Policies

4.1. Late Submissions:

Late papers and projects and make-up exams will not normally be permitted. I will give appropriate consideration to documented emergencies, but such arrangements must be made prior to the due date in any situations where the conflict is foreseeable. Extensions to due dates will not be granted simply to allow “porting” from
one system to another. “But I had it working on my home PC!” is not an acceptable excuse. In a similar vein, anyone who has achieved junior-level or higher status in a CS program should by now be aware that the network and workstations generally get overloaded just before assignments are due, and that machine crashes and downtime are a fact of life in this field. Students are expected to plan for these problems. Except in extreme cases, these are not grounds for extension of a due date.

4.2. **Attendance:**

Attendance at classes is not generally required, but students are responsible for all material covered and announcements made in class. Consequently, if you are going to miss class, be sure to get notes, handouts, etc., from another class member.

4.3. **Academic Honesty:**

Everything turned in for grading in this course must be your own work. The instructor reserves the right to question a student orally or in writing and to use his evaluation of the student’s understanding of the assignment and of the submitted solution as evidence of cheating. Violations will be reported to the Honor Council for consideration for punitive action.
By CS Dept. policy, students found to be in violation of this rule will, at the very least, receive a failing grade in the course and may be subject to stiffer penalties. Students who contribute to violations by sharing their code/designs with others are subject to the same penalty.

This policy is not intended to prevent students from providing legitimate assistance to one another. Students are encouraged to seek/provide one another aid in learning to use the operating system, in issues pertaining to the programming language, or to general issues relating to the course subject matter. Student discussions should avoid, however, explicit discussion of approaches to solving a particular programming assignment, and under no circumstances should students show one another their code for an ongoing assignment, nor discuss such code in detail.

### 4.4. Grading:

Assignments: 30%
Midterm Exam: 30%
Final Exam: 40%

It is my general policy that, should a student perform significantly better on the final than upon the midterm, or should a student have one assignment grade that
is significantly lower than the rest, to waive that single low grade (adjusting the percentages of the remaining grades accordingly).

**4.5. Adding/Dropping/Incompletes**

Students may add the course or may drop with no record of it on their transcript up until 9/2. After that, students may drop the course, receiving a “W” grade on their transcript up through 10/21.

Students who drop a course are requested to notify the instructor, even when the instructor’s permission is not required.

A grade of “I”, for Incomplete, may be given in exceptional circumstances beyond the student’s control to indicate that work remains to be completed by the student after the end of the semester. Such work must be completed before the end of the following semester, or the grade automatically becomes an “F”. The student must contact the instructor to discuss the possibility of an “I” grade and to negotiate the nature and schedule of the work to be completed.

**4.6. Topics**

1. I. Language Components
Section 4: Course Policies

(a) Classification
(b) Translation & Syntax

2. Imperative Languages (Pascal, C, C++, Modula 2, Ada)
   (a) Statements
   (b) Data Types and Structures
   (c) Procedures and Activations

3. Modularity & Object-Oriented Programming (C++, Modula 2, Ada, Java)
   (a) Modules
   (b) Object-Orientation

4. Functional Programming
   (a) ML
   (b) LISP

5. Logic Programming (Prolog)

6. Little Languages — Scripting (bash, awk, Tcl, Tk, Expect, Perl)

A more detailed schedule may be found at