CS164, Spring 2015: Course Information

Personnel

Instructor:

Paul N. Hilfinger
787 Soda Hall
Phone: 642-8401
Email: Hilfinger@cs.berkeley.edu, cs164@cs
Office Hours: TBA

Teaching Assistants:

Achal Dave
Email: achal@berkeley.edu
Office hours: M 4-5 in 651 Soda and W 2-3 in 283E Soda.

Vedant Kumar
Email: cs164-td@imail.eecs.berkeley.edu
Office Hours: Tu 10-11, Th 10-11 in 411 Soda

Omar Ramadan
Email: cs164-tc@imail.eecs.berkeley.edu
Office Hours: F 2-3 in 411 Soda

Purpose of this Course

CS164 is an introduction to the design of programming languages and the implementation of translators for them. In the process, we'll do some general exploration of programming language design and its implications for implementation, and look at a dialect of at least one particular language, which this semester is Python.

One goal of this course is to explore the structure of programming languages and to consider alternatives to familiar programming language features. We'll also study the problem of translating programming languages into machine-executable forms, using Python as a concrete example of a language to be translated, and the assembly language of the Intel ia32 family (used in PCs and some of our Solaris workstations downstairs) as a concrete example of a target machine. We study language translation first to learn some of techniques used that are useful for many programming problems outside of language translation, second to gain a better intuitive feel for the tools we use when programming and the costs of the programs we write, and third (possibly most important) to gain experience with the engineering problems associated with building and validating a substantial piece of software.

Prerequisites

Please do not take this course unless you have the prerequisites: CS61B and CS61C, or equivalent courses (at a junior college, for example). You should be familiar with Java or C++. You need not be familiar with Python.

Reference Material

There is no required textbook for this course, since none is altogether satisfactory, and all tend to be ridiculously expensive. Officially, we'll rely on notes that I provide (on paper and on the web), but many people feel more comfortable if they can supplement this information with other treatments. You might want to take a look at the following books:

- *Compilers: Principles, Techniques, and Tools* by Aho, Sethi, and Ullman—also known as "The Red Dragon Book". You won't need the latest edition.
- At least one student recommends *Thinking in C++*, by Bruce Eckel, which you can download by clicking here.
- *Python Essential Reference (3rd edition)* by David M. Beazley. This is also on-line (see the bar on the left) and at bookstores, on-line and otherwise.

Course Work

The major work in the course consists of building a system to translate programs written in a dialect of the popular scripting language Python. The system will be divided into modules, with each module constituting an assignment. The implementation language is C++ (!). In addition to the project, there will be a midterm, a final examination, and some ordinary written homework.

Collaboration

You may work on the various modules in teams of two or three. One item on the secondary agenda of this course is to learn something about collaboration on software projects—an important fact of life in the real world. Collaborating with someone often encourages more attention to your programming technique and gives you practice in one of the social aspects of computing that university courses must generally ignore. As an extra bonus, working in groups substantially reduces the strain on our computing and grading resources. Each team will submit one module and each member of the team will receive the same grade for that module. The exams may contain questions about the modules, so it is extremely unwise to allow your partner to do all the work. Although the Staff (elsewhere known as "we") will be happy to give advice, it is the responsibility of partnerships to work out their own disputes. Partnerships apply only to the project assignments, *not* to tests or other written homework. Students taking the course Pass/Fail may *not* take graded students as partners.

I encourage collaboration among teams, as long as it is restricted to questions of strategy as opposed to actual program text. Each partnership or individual is responsible for making it clear that their work is really theirs. Except for general-purpose utilities (e.g., a doubly-linked list package) you should not generally borrow other people's code. For anything you do borrow (including ideas) you must give proper credit in comments. Naturally, over-dependence on others' ideas will adversely affect your grade on a module, but failure to give credit where credit is due falls under the heading of "cheating."

Cheating (also known as plagiarism) is the presentation of someone else's work as your own. It will not be tolerated. Violators will fail the course and will be reported to the appropriate disciplinary agency. It is not possible for only one member of a partnership to cheat on a project assignment; any transgressions on your part will reflect equally on your partner.

Grading

In an attempt to cut down on cutthroat competitiveness and to make your grade a bit more meaningful, I use an absolute grading scale. Your letter grade will be determined by the total number of points on all assignments: A for 160–200 points, B for 130–160, C for 100–130, and D for 75–100, with plusses and minuses awarded to (roughly) the top and bottom third of each grade. These divisions are based on past classes' performances. Out of the total 200 points, the final counts for 50, the two additional tests for 20 apiece, and other written homework and project assignments total 110.

Ordinary homework doesn't get many points, because I see it mostly as a chance for exercise. You will be given full credit for handing in a solution in which you demonstrate that you have made a substantial effort on each problem. Since solutions will be available after the due date, late homework *won't* be graded.

Unless otherwise indicated, you should hand in all assignments electronically by the deadline given on the assignment. I'll penalize a project $5N/12$ percent for each N hours it is late, rounded off in some unspecified fashion. During the course of the semester, however, we'll give you a *total* of four late days (96 hours) for free. For the first assignment, you'll have up to 32 hours of lateness for free. For the second assignment, you'll get another 32 hours, plus any you have left over from the first. Similarly for the last assignment. No, you cannot get late hours on credit; we won't move any of your Project 2 hours to Project 1. Needless to say, you are not obliged to turn in *any* of them late, and it is advisable never to be more than one day late. In general, it is the final submission (or partial submission) of something that counts, so making a change one day after a deadline is one day late. We will often forgive lateness for trivial changes, and we can ignore inadvertant late submissions entirely. Your best procedure, if you want to make a small change in an assignment and aren't sure whether we'll let you do it without lateness penalty, is to submit the change and *then* ask us. We can always undo a submission if it turns out to have been the wrong thing to do.

Page was last modified on Mon Jan 26 13:37:47 2015.
Address comments and questions to cs164@cs.berkeley.edu