CS 301 Languages and Compiler Design I - Winter 1999

Instructor:
Andrew Tolmach
PCAT 128A
725-5492
email: `apt@cs.pdx.edu`
Office Hours: MW 2-3 & by appt.

Course home page: `http://www.cs.pdx.edu/~apt/cs301`

Description

CS301/302 studies, in parallel, the design and implementation of programming languages and the design and implementation of compilers. The course is centered around a substantial programming project: implementing a complete compiler for a realistic language. We will study formal methods for specifying the syntax and semantics of languages, and, in some cases, use tools based on these methods to help build the compiler. We will examine the structure and capabilities of programming languages with an emphasis on the demands of efficient implementation. A detailed list of lecture topics may be found below.

Prerequisites

Formal prerequisites are CS202, 252, and 300. Substantial experience with the C or C++ programming language is *essential*. Experience with the Unix operating system, familiarity with a block-structured language such as Pascal or Modula-2, and some prior exposure to basic automata theory will be very helpful. For CS302, some experience with machine-language programming will be extremely desirable.

Texts

The required textbook is Aho, Sethi, and Ullman, "Compilers: Principles, Techniques, and Tools," Addison-Wesley, 1986. Additional readings from other sources may be handed out in class from time to time.

Another useful book to know about is Levine, Mason, and Brown, "lex & yacc," 2nd ed., O'Reilly & Associates, 1992.

Lecture notes will be available electronically in postscript and html formats via the course home page.

Project

Over the course of CS301/302 you will write a complete compiler for the PCAT programming language, a distant relative of Pascal. A description of the language will be handed out in class.

Your compiler must be written in ANSI C. C is described in B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd edition, Prentice–Hall, 1988, and in numerous other reference books.

Exams

There will be one mid-term and a final exam. Both are *closed-book*. Exams will cover topics from lectures and readings, emphasizing material that is not directly relevant to the programming project. Not all the material is covered in the readings, so lecture attendance is important. Exams are scheduled in advance; unless prior arrangements are made, a grade of zero will be recorded for missed exams.

Homework problems may be assigned from time to time as an aid to help you study for the exams; they will not be collected or graded.

Grading

Approximately 1/2 on programming assignments and 1/4 on each of 2 exams (including the final).

Programming assignment grades will be determined primarily by observing program behavior on test inputs. Computer programming being what it is, this policy means that "small" errors in your code can have a large effect on your grade. Sample test inputs and a correctly-behaving executable will be provided for you to compare your program against. It is your responsibility to apply these tests and others of your own devising before submitting your assignment. We will apply some non-public tests to your programs as well.

Computing Facilities

You may develop your project solutions on any machine and operating system you like, but the code you hand in must work using the GNU `gcc` compiler and associated tools on the CS department's Sparc machines, which run Solaris (a variant of Unix). Because several assignments strongly encourage (though do not absolutely require) the use of the Unix tools `lex` (or `flex`) and `yacc` (or `bison`), you may be seriously handicapped by not using a Unix-based machine to do them. Linux machines should work fine. Also, each assignment will come with a working solution that you can execute; the executables will only work in the Sun/Solaris environment. You should have been given an account on the CS machines automatically by virtue of registering for this course; to obtain your password, take your id to the PCAT lab. Files associated with assignments will be placed in directory `/u/cs301acc` on the CS Suns, and will also be available from the course web page.

Submitting Programs

Programs should be submitted by `mail`ing a "bundle" containing the relevant files to `cs301acc@cs.pdx.edu`. Bundles can be created by the Unix `shar` command, which combines the files given by its arguments into a single shell script that, when run, recreates the files. For example, you could submit the files `makefile`, `lex.c`, and `main.c` as follows:

```
shar makefile lex.c main.c | mail cs301acc
```

Do not send mail other than program submissions to `cs301acc`, as it is not regularly read.

Mailing List

Important information will be distributed throughout the term via a Majordomo list called `cs301list`. You can subscribe to this list by sending a mail message to `majordomo@cs.pdx.edu` with no subject line and a message body consisting of the single line

```
subscribe cs301list
```

Please mail questions to the instructor directly (at `apt`) rather than to this list; the instructor will copy mail of general interest to the list.

Schedule

All dates, including assignment deadlines, are tentative.

Date		Asgn	ASU Reading	Topics
Jan	4		1	Introduction
	6	1	2.1–7,2.9	Grammars; Abstract syntax; Interpretation
	11			Language design; Historical survey
	13	2	3.1–4	Lexical analysis; Regular expressions
	18			**Martin Luther King, Jr. Holiday – no class**
	20	1*	3.5–8	Finite automata; Scanner generation
	25		4.1–3	Syntactic analysis; Grammars
	27		4.4	Top-down parsing; Recursive descent
Feb	1			Table-driven LL parsing
	3	3 2*	4.5	Bottom-up parsing
	8			**Midterm Exam** (in class)
	10		4.9	Parser generators
	15		5.1–5	Syntax-directed translation
	17			Data types
	22	4	6.1,6.3,8.2	Type declarations and equivalence
	24	3*	6.2,6.4–5	Type checking and coercion
Mar	1			Abstract data types and modules
	3			Object-oriented programming concepts
	8			Functional programming concepts
	10	4*		Semantics of programming languages
	15			**Final Exam** (starting at **3:30 pm**)

All indicated readings are from the Aho, Sethi, and Ullman textbook. Additional assigned readings may be handed out from time to time. Assigned selections should be read *before* the associated date.

Assignments

Programming assignments, listed below, are distributed on the dates the assignment number appears in the schedule above and are due when the starred number appears.

1 simple interpreter
2 lexical analysis
3 parsing
4 type checking

Rules

Programs are due *by 3:30 p.m.* on the specified due date, i.e., just *before* class. Late programs are *not* accepted except in extraordinary circumstances, and then preferably by prior arrangement. The project deadlines are there to help you, by forcing you to keep up during the term.

All programming assignments must represent your own, individual work. It is permissible to discuss the assignment with other students, but not to discuss the solution (although you can consult the tutors for help in debugging). *Do not, under any circumstances, copy another person's program and submit it as your own.* Writing code for use by another or using another's code in any form (even with their permission) will be considered cheating, the penalties for which are described in detail in the CS Department's booklet *Undergraduate Programs 1998-1999*. In particular, cheating will result in an automatic F for the assignment in question, and a letter to your departmental file.