Compilers and Interpreters

CS 4352

Spring Semester, 2005

Syllabus

http://www.cs.utep.edu/nigel/compilers/

Time and Location
15:00 – 16:20, Tuesdays and Thursdays
Computer Science 321

Instructor
Nigel WARD
nigel @ cs.utep.edu
Computer Science, Room 206
(915) 747-6827

Course Goals:

- Improve programming skills
- Improve systems integration skills
- Become a wiser user of compilers and interpreters
- Reinforce basic computer science concepts
  - data structures
  - programming langugage features
  - assembly language and computer architecture
  - formal language and automata theory
  - operating systems
- Learn how to build compilers and interpreters
  - overall
  - algorithms, data structures, useful techniques
  - development techniques
- Acquire Unix skills

Prerequisite
CS 3350 (Automata, Computability and Formal Languages)

Textbook
Programming Language Processors in Java: Compilers and Interpreters, by David A. Watt and Deryck F. Brown, Prentice Hall, 2000. (henceforth WB)
(web site at `http://www.dcs.gla.ac.uk/~daw/books/PLPJ/`)

We will follow the textbook fairly closely.

Please bring it to class whenever possible.

It is important that you read the relevant sections before each class, roughly following the schedule below.

There may also be supplemental readings, handed out in class.

Assignments
In this class you will constuct an interpreter and a compiler for a small langugage. This will be done via a large number of relatively small assignments.

The assignments will be cumulative, however periodically during the semester the instructor will distribute good solutions so that no team gets hopelessly behind.

Most assignments may be done either individually or in pairs. Some assignments will be done partly in class.

Assignments are due at the start of class. Assignments more than one minute late will be given half credit. Assignments more than one day late will be penalized further and only accepted with previous permission from the instructor.

Assignments may be done in any language. However a basic knowledge of Java will be invaluable for understanding the book and some of the provided code.

Assignments are to be turned in as print-outs, typically including a screendump and the code written.

Assignments will be graded primarily on performance and on quality of the code, including comments and documentation. Extra credit will be given to the first person to inform the instructor of any bug in the assignments.

Tests:
There will probably be two tests, tentatively February 10 and March 17.

There will be a final examination, probably 13:00–15:45, Thursday, May 5.

Grading:
The weighting will be approximately: Final Exam 35%, Assignments 30%, Tests 25%, Quizzes 5%, and Class Participation 5%.

Office Hours:
Fridays 13:15–14:15 in my office, or by appointment, or whenever the door is open. Come with any questions, or just to chat.

Tentative Schedule of Readings and Assignments

Lecture Topics	Readings	Assignments
Day 1: Course Overview	*WB 1.1, 1.2, 2.2*	P1: portability considerations [2 hours]
Day 2: Interpretation	*WB 8.1, 2.3*	I1: build a simple interpreter [1 hour]
Day 3: Porting Code (Prof. Bell)	*WB 2.1*	
Day 4: Bootstrapping	*WB 2.4-2.5*	I2: add simple loops and batch mode [2 A]
Day 5: More Bootstrapping	*WB 2.6-2.7*	I3: paintball pseudocode [2 A]
Day 6: Tokenizing/Scanning	*WB 4.1, 4.5*	
Day 7: Scanning; Unix		I4: add variables [1]
Day 8: Regular Expressions, BNF, lex		L1: using lex [2]
Day 9: More Unix; Lab Time		U: using regular expressions and shell scripts [2]
Day 10: Test 1; Unix Lab Time		
Day 11: Language Specification		I5: language design: conditionals [1]
Day 12: Compiler Parts Overview	*WB 1.3, 1.4, 3.1*	I6: implement conditionals [2]
Day 13: Compiler Organization	*WB 3.2-3.3, Apx. B,C*	T: understand Triangle and TAM [2]
Day 14: Grammars, Recursive Descent P.	*WB 4.2, 4.3*	C1: extend the scanner (underscore in identifiers) [1]
Day 15: Table-Driven Parsing		
Day 16: AST Construction Overview	*WB 4.4, 4.6, 9.2*	C2: design a pretty-printer [1]
Day 17: Contextual Analysis	*WB 5.1*	C3: add precedence to the grammar [2]
Day 18: Type Checking	*WB 5.2, 5.3, 5.4*	C4: new data type: design [1]
Day 19: Expression Evaluation	*WB 6.1, 6.2*	C5: new data type: front end [2]
Day 20: Test 2		
Day 21: Storage Allocation	*WB 6.3, 6.4*	O1: stack allocation analysis [1]
Day 22: Scope, Routines	*WB 6.5*	C6: new data type: back end [3]
Day 23: Code Selection	*WB 7.1, 7.2*	O2: code generation analysis [1]
Day 24: Storage Allocation, Again	*WB 7.3*	C7: overload an operator ...
Day 25: Control Structures	*WB 7.4, 7.5*	... or add a new control structure [3]
Day 26: Code Optimization	*WB 9.3, Parsons Ch. 6*	R1: Roboland Compiler Runtime Design [2]
Day 27: More Code Optimization		R2: Roboland Compiler I1-Level Implementation [4]
Day 28: Object-Oriented Languages	*WB 6.7*	R3: Roboland Compiler I2-Level Implementation [2]
Day 29: Heap Management	*WB 6.6*	R4: Complete Roboland Compiler [2]
Day 30: Review	*WB 9.1, 9.2*	