SYL-410-2014C

CS 410 - Compiler Construction

West Virginia
fall semester 2014
August 18, 2014
syllabus 1.0

Course location:	107 ERB, Evansdale Campus
Course times:	Tuesdays and Thursdays, 2:00-3:15
Course Format and Credit Hours:	3 hours lecture 3 credit hours
Prerequisites:	CS 110 Introduction to Computer Science CS 220 Discrete Mathematics CS 310 Principles of Programming Languages
Instructor:	Frances L. Van Scoy 304.293.0960; 827 ESB frvanscoy@mail.wvu.edu
Office Hours:	9:30-10:45 and 12:30-1:45 Tuesday and Thursdays 827 ESB
Course Objectives:	Theory and practice of the construction of programming language translators; scanning and parsing techniques, semantic processing, runtime storage organization, and code generation; design and implementation of interpreter or a compiler by students.
Expected Learning Outcomes:	• Understand Chomsky hierarchy and its relationship to compiler construction • Construct compiler components
Required Texts:	Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman Compilers Principles, Techniques, and Tools, 2nd edition Addison-Wesley, 2007 and Doug Brown, John Levine, Tony Mason lex & yacc, second edition O'Reilly Media, 1992 available via WVU Libraries' subscription to Safari Books Online and Smalltalk-80: The Language and its Implementation By Adele Goldberg and David Robson; Xerox Palo Alto Research Center ISBN 0-201-11371-6. 344 pp. 1983 avalable for legal free download (32 MB) at http://stephane.ducasse.free.fr/FreeBooks/index.html

"Compiler construction" is one of the earliest success stories for computer science. That is, there was a real problem to solve--how to translate notation easily written and understood by humans into strings of binary digits understood by a computer--for which a rather elegant body of theory provided much of the solution.

There are two organizing themes for this course.

the phases of a compiler:

scanner, parser, symbol table manager, intermediate code generator, code generator

the Chomsky hierarchy:

regular expressions, context free, context sensitive, phrase structure

This course is a mix of formal language theory, construction of a software system using lex and yacc, pragmatic programming details, comparative programming languages, and natural language processing..

Grading:	scanner assignment		7%
	parser assignment		7%
	term project		30%
	design	6%	
	scanner	6%	
	parser	6%	
	running draft program	6%	
	final project	6%	
	Gradiance assignments, 12 @ 4%		48%
	Reflection essays 4 @ 2%		8%

Grading scale:	A	90-100%
	B	80-89%
	C	70-79
	D	60-69%
	F	0-59%

48% of the course grade is based on reading assignments in the Aho text book evaluated by 12 weekly assignments via Gradiance. The first one is due 11:59 pm, Friday, August 29.

Go to www.newgradiance.com to set up a free account.

Your class token is **4994C47C.**

Policies

Attendance is a not a factor in computing the grade, EXCEPT:
(1) you are responsible for all material presented, handouts distributed, announcements made, etc., in class
(2) if you miss a test or exam without prior approval, your grade on that test or exam will be 0

If you copy someone else's answers on a test, knowingly allow someone to copy from your test, turn in someone else's work as your own, or cheat in any other way, you can receive a failing grade in this course. There may also be further disciplinary measures. The penalty will always be more severe than a failing grade in the test or assignment involved.

I expect to send frequent messages to the class via email using the MIX system. You should either check your MIX email account daily or forward your MIX email to an account where you check mail daily.

Course outline

0. Course overview (1)
    a. block diagram of a compiler
    b. Chomsky hierarchy

1. Lexical analyzer (6)
    a. Problem: breaking text into words
    b. Linux scripts, pipes and filters, sed
    c. lex
    d. regular expressions
    e. lexical analyzer
    f. left linear grammars and right linear grammars
    g. Snobol and Icon pattern matching
    h. NLTK

2. Syntax analyzer (7)
    a. Problem:
    b. Backus Nauer form and Extended Backus Nauer Form
    c. Context free grammars
    d. yacc
    e. LL parsing
    f. LR parsing
    g. Pushdown automata

3. Symbol table (2)
    a. Problem: Curveship spins
    b. Example: Pascal "Hunt the Wunpus" declarations

4. Intermediate code generator (7)
    a. Problem:
    b. yacc revisited
    c. Example:
    d. Context sensitive grammars
    e. 2-stack pushdown automata and linear bounded automata
    f. Smalltalk bytecode generation
    g. Type checking
    h. Ada type checking
    I. Run-time environments

5. Compiler back-end (2)
    a. Code Generation
    b. Optimization

6. Implementation examples (1)
    a. Curveship implementation in Python

7. Summary (0)

Schedule

Lecture	Date	Title	Reading	Due (11:59 pm Saturday)
1	"August 19	Course overview: compiler components, Chomsky hierarchy	Aho 1, 2	
2	"August 21	Lexical analyzer: motivation; problem: breaking text into words; Linux scripts, pipes, filters, sed	Aho 3	
3	"August 26	lex	Brown 1,2, 6	
4	"August 28	regular expressions		
5	"September 2	lexical analyzer		
6	"September 4	left linear grammars and right linear grammars; finite state automata		term project: design
7	"September 9	Snobol and Icon pattern matching; NLTK		
8	"September 11	Syntax analyzer: recognizing sentences; BNF and EBNF	Aho 4	lex assignment
9	"September 16	context-free grammars	Aho 5	
10	"September 18	yacc	Brown 3, 4, 5, 7, 8, 9	Reflection 1: class meetings 2-7
11	"September 23	LL parsing	(Aho 4)	
12	"September 25	LR parsing 1		term project: scanner
13	"September 30	LR parsing 2		
14	"October 2	pushdown automata		yacc assignment
15	"October 7	Symbol table: Curveship spin		
16	"October 9	Pascal "Hunt the Wumpus" symbol table		
(fall recess)	"October 14			
17	"October 16	Context-sensitive grammars		term project: parser
18	"October 21	2-stack pushdown automata		
19	"October 23	Intermediate code generation: ; yacc revisited	Aho 6	Reflection 2: 8-14

20	"October 28	Intermediate code for common language structures		
21	"October 30	Smalltalk bytecode generation	Goldberg, 26-30	Reflection 3: 15-18
(election day)	"November 4			
22	"November 6	Type checking; Ada example		
23	"November 11	Run-time environments	Aho 7	
24	"November 13	Code generation	Aho 8	
25	"November 18	Optimization	Aho 9-12	
26	"November 20	Implementation: Curveship implementation example		term project: a functioning program (that may be missing some functionality)
(fall recess)	"November 25			
(Thanks-giving)	"November 28			
27	"December 2	Project presentations 1		Reflection 4: 19-26
28	"December 4	Project presentations 2		
29	"December 9	Project presentations 3		term project: source code, documentation, presentation material