

Compilers - CS 431
Fall 2008
Credits: 4 hours

Instructor: David Furcy
Email: furcyd@uwosh.edu

Office: Halsey 220
Phone: 424-1182

Class Meetings: 3:00-4:30PM on Mondays and Wednesdays in HS 208
3:00-4:00PM on Fridays in HS 202

Office Hours: **MTuWThF** 8:00-9:00AM
Feel free to drop in at any other time when my door is open.
Alternatively, to ensure that I have enough time to answer your questions, I strongly encourage you to make an appointment.

Prerequisites: CS 331 and Math 122, each with a grade of C or better.

Required Textbook: None

References:

- Class handouts. **TAKE NOTES** on and about them. Handouts that are not liberally saturated with your own notes will likely prove useless when you need them most!
- Course web page at http://www.uwosh.edu/faculty_staff/furcyd/cs431/

Tests: Exam #1: Thursday, October 23rd, 5:00-7:00PM
Exam #2: Thursday, December 11th, 5:00-7:00PM

Exams are scheduled at night to circumvent the time limit imposed by in-class exams. Please block these time slots at the beginning of the semester. The procedure and criteria of eligibility for making up a missed exam are described below.

If you have special needs (or ABSOLUTELY cannot make it to one of the above night exams), please come and talk to me RIGHT AWAY so I can accommodate your needs.

Topic Coverage:

- Syntactic analysis
 - Lexical analysis
 - Using scanner and parser generators
 - Context-free grammars and parsing
 - Top-down parsing: Recursive-descent (or predictive)
 - Bottom-up parsing: LR
 - ASTs: Abstract syntax (or parse) trees

- Semantic analysis
 - Symbol tables
 - Type checking
- Runtime organization
 - Activation records and run-time stacks
 - The JVM
- Code generation
 - Java bytecodes
 - Instruction selection
- If time permits, one or more of the following topics:
 - Liveness analysis
 - Register allocation
 - Garbage collection
 - Loop optimizations
 - Pipelining and scheduling
 - The memory hierarchy

Learning Outcomes: At the conclusion of the course, the student will be able to:

- Describe the compilation process, including the roles of the scanner (lexical analyzer), parser, semantic analyzer, intermediate code generator, optimizer, and code generator
- Define a deterministic finite state automaton (DFA)
- Explain how a DFA can be used to recognize a token
- Discuss the relationship between DFA's and regular expressions
- Define the lexical structure of a programming language using regular expressions
- Implement a lexical analyzer using a scanner generator
- Explain how a context-free grammar can be used to express the syntactical structure of a programming language
- Compare the respective capabilities of regular expressions and context-free grammars
- Construct a parse tree for an expression defined by a context-free grammar
- Discuss the differences between LL, LR, and LALR grammars
- Trace the actions of a recursive descent parser in processing an expression defined by an LL grammar
- Trace the actions of a top-down table-driven predictive parser defined by an LL grammar
- Trace the actions of a bottom-up table-driven parser defined by an LR grammar
- Define the syntactical structure of a programming language using a context-free grammar
- Implement a parser using a parser generator
- Explain how syntax-directed translation is used to augment a context-free grammar with translation rules
- Define an abstract syntax tree
- Compare abstract syntax trees to parse trees
- Apply syntax-directed translation to be able to generate abstract syntax trees

- Explain the role of semantic analysis and its relationship to type checking and scoping issues in programming languages
- Discuss the role played by symbol tables in semantic analysis
- Compare different implementation strategies for symbol tables
- Implement a semantic analyzer for a programming language
- Explain what an activation record (stack frame) is and discuss its role in function calls and parameter passing
- Explain the role played by intermediate code generation and compare a compilation process that uses it to one that does not
- Implement a code generator for a programming language
- Test each of the implementations above with an appropriately designed set of test cases
- Collaborate with team members, apply sound software engineering principles, design patterns, and robust coding techniques, to successfully complete a large, software project

Course Grading Policy: Your final grade for this course will depend on a semester-long programming project and 2 exams. The goal of the project is to write a compiler for (a subset of) the Java language. Like *javac*, your compiler will translate Java programs into bytecodes to be executed by the Java Virtual Machine. The project will be broken down into 5 assignments. Each assignment and exam will be graded on a scale from 0 to 100. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Assignment 0	5%
Assignment 1	10%
Assignment 2	15%
Assignment 3	15%
Assignment 4	15%
Exam #1	20%
Exam #2	20%

Finally, your letter grade for the course will be computed using the following mapping:

Numerical Score	Course Grade	Numerical Score	Course Grade
≥ 92	A	≥ 72	C
≥ 89	AB	≥ 69	CD
≥ 82	B	≥ 60	D
≥ 79	BC	< 60	F

I will be glad to discuss any questions you may have about grades. However, make sure to bring them up right away, upon return of each graded assignment or exam. Last minute requests after the final exam will not be entertained.

Attendance and Participation: You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me outside of class to discuss any questions you may have, and to have completed the programming assignments on time.

It is hard to imagine how a student could do well in this course while missing classes, attending them unprepared, or not participating.

On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage** you to **ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment. Send me email or give me a call to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available before an assignment is due or in the few days preceding an exam.

Late Submissions: I will describe the submission procedure for your assignments when the time comes. However, let me point out right away that each assignment will come with a deadline (day and time) after which any submission is considered late, **with no exception**. The penalty for late submissions is computed as follows:

Turned in	Penalty
on due date, after deadline	10%
one day late	20%
two days late	40%
three days late	60%
four days late	80%

Note that submissions that are more than 4 days late will receive zero points. Late submissions can easily be avoided by starting to work on the assignment right away and asking me questions early if you get stuck.

The penalty for late submissions can be waived in **only one** scenario, namely if you give me a signed note from the attending physician or a written justification for the extension from the Dean of Students Office. If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give me a valid justification (see above) ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester.

Bonus points: Since late submissions are penalized, it is only fair to reward early ones as well. Therefore, if you submit an assignment at least one full day early (and up to four days early), you will earn 3 bonus points for every full 24-hour interval between your submission time and the deadline time. Thus a maximum of 12 bonus points may be earned for each assignment. Furthermore, there will be other opportunities for students to earn bonus points throughout the semester.

Collaboration versus Cheating: All submissions must be the work of all, and only, the students on the team. While it is acceptable to discuss the assignments with others, you must submit your own work. You may **not look at** or “borrow” any piece of code of any length from anyone else, unless you can live with a zero and the other potential academic sanctions of cheating. Check out the UWO Student Discipline Code (UWS 14) at <http://www.uwosh.edu/dean/conduct.htm> for details.

Final Note: I expect every committed and hardworking student to do well in this course. I am looking forward to a fun and rewarding semester together. Given my high standards, I could not be more satisfied than if everyone earned an A in this course.