CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

www.cse.buffalo.edu/faculty/alphonce/SP22/CSE443
PROTECT THE UB COMMUNITY—AND YOURSELF

WEAR A HIGH-QUALITY MASK

Masks are required inside campus buildings, on UB transportation and in large outdoor gatherings.

Masks should fit snugly over your nose and mouth and provide high filtration.

For more information, visit buffalo.edu/coronavirus.

Together, we are stopping the spread.
Roadmap

- Syllabus: posted on website
- Course overview
- Course structure
BUILD

A

COMPILER!

What?
Why?

- Deeper understanding of languages
- Become a better programmer
- Learn how to build tools
- Build special-purpose languages (DSLs)
- Theory meets practice
- High-level meets low-level
How?

- That's the rest of the course!
Assessment plan

- Homework (20%)
  - four - due dates on web site schedule
  - mix of final exam prep and project support

- Project (60%)
  - four milestones (5% each) before final submission (40%)
  - team-based

- Final Exam (20%)
  - during final exam period
  - sample questions give out the last week of classes
# Learning outcomes

<table>
<thead>
<tr>
<th>Learning outcome</th>
<th>Instructional methods</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify and describe the function of the major phases of a compiler.</td>
<td>Lecture-based instruction</td>
<td></td>
</tr>
<tr>
<td>Define formally the grammars used in the front end of a compiler, their</td>
<td>Hands-on lecture activities</td>
<td>HW, EX</td>
</tr>
<tr>
<td>application in the front end, and techniques for parsing such grammars.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluate (compare and contrast) different intermediate representations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explain the compiler’s role in creating and managing run-time environments.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explain and evaluate (compare and contrast) different approaches to code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>generation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify and explain the applicability and operation of code optimizations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build both the front and back ends of a compiler.</td>
<td>Lecture-based instruction</td>
<td>PROJ</td>
</tr>
<tr>
<td>Collaborate effectively as a member of a software development team.</td>
<td>Team project w/team-faculty meetings</td>
<td>PROJ</td>
</tr>
</tbody>
</table>
Teams & Meetings

- Form teams no later than Tuesday next week (after add/drop)
- Teams must be of size 3 or 4.
- Teams must set up mandatory weekly meetings with me (~50 minutes)
- One member of each team must make a private post in Piazza with the UBIT of each person on their team.
- All code must be maintained in private git repo hosted on GitHub - I will help teams set this up at first meeting via GitHub Classroom; don't set this up on your own before then.
Goal: build a compiler

source program

<table>
<thead>
<tr>
<th>↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>executable</td>
</tr>
</tbody>
</table>
Phases of a compiler

source program

Figure 1.6, page 5 of text
Phases of a compiler

source program

executable

Figure 1.6, page 5 of text
Setting the stage
Deep understanding - ex 1

name vs identifier vs variable
name

y.x

identifier

x

refers to

variable location in memory
void foo(void) {
    int x = 0;
    printf(x);
}

void bar(void) {
    double x = 3.8;
    printf(x);
}
Deep understanding - ex 1

```c
struct Pair {
    int x;
    int y;
};

void bar(void) {
    struct Pair r, s;
    /* ... */
}
```
Deep understanding - ex 1

```c
int f(int x) {
    if (x == 0) { return 1; }
    else { return x * f(x-1); }
}
```
variables in distinct scopes, variables in distinct records/objects, or variables in distinct function invocations
Deep understanding - ex 2

order of evaluation

Does source code completely determine order of evaluation/execution at machine language level?
Deep understanding - ex 2

\[ a + b \times c; \]

What is the order of evaluation?
Deep understanding - ex 2

\[ f(0) + g(0) \times h(0); \]

What is the order of evaluation?
Deep understanding – ex 2

\[ f(0) + f(0) \times f(0); \]

What is the order of evaluation?
In most languages the result will be consistent with the evaluation of
\[ a + (b \times c) \]
Deep understanding - ex 2

\[ a + b \times c; \]

Order of operations is important here, but order of evaluation of the variables \(a, b,\) and \(c\) is not (as long as they are evaluated before they are needed).
Deep understanding - ex 2

```
a++ + a++ * a++;
```

Order of operations is important, as is order of evaluation of the variables if there are side effects!