Programming Language Design and Implementation

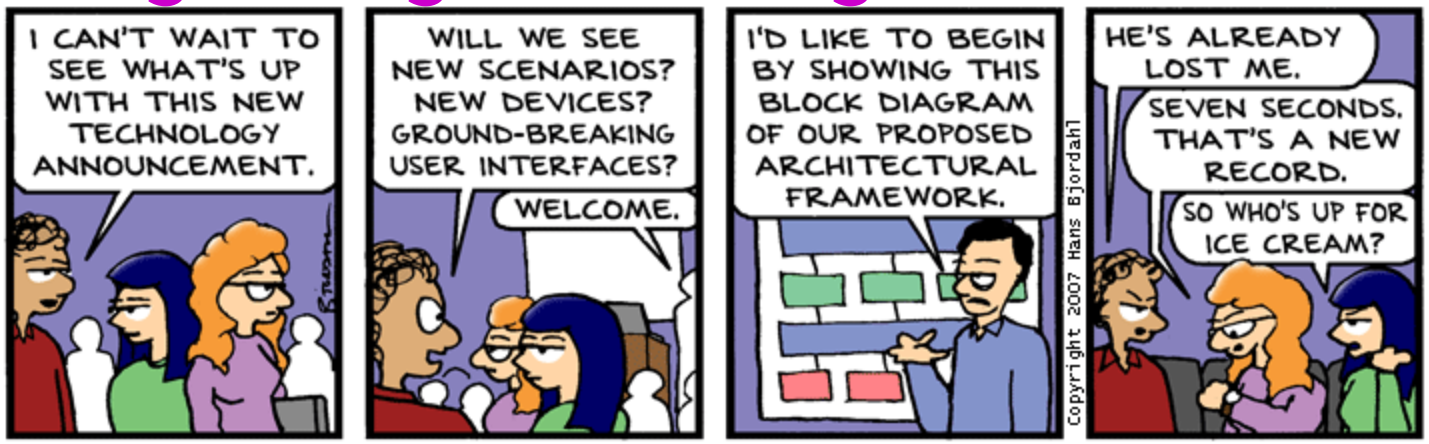
Wes Weimer

TR 5:00 to 6:15

OLS 009

Cunning Plan

- Who Are We?
  - Wes, Zak Fry
- Administrivia
- What Is This Class About?
- Brief History Lesson
- Understanding a Program in Stages



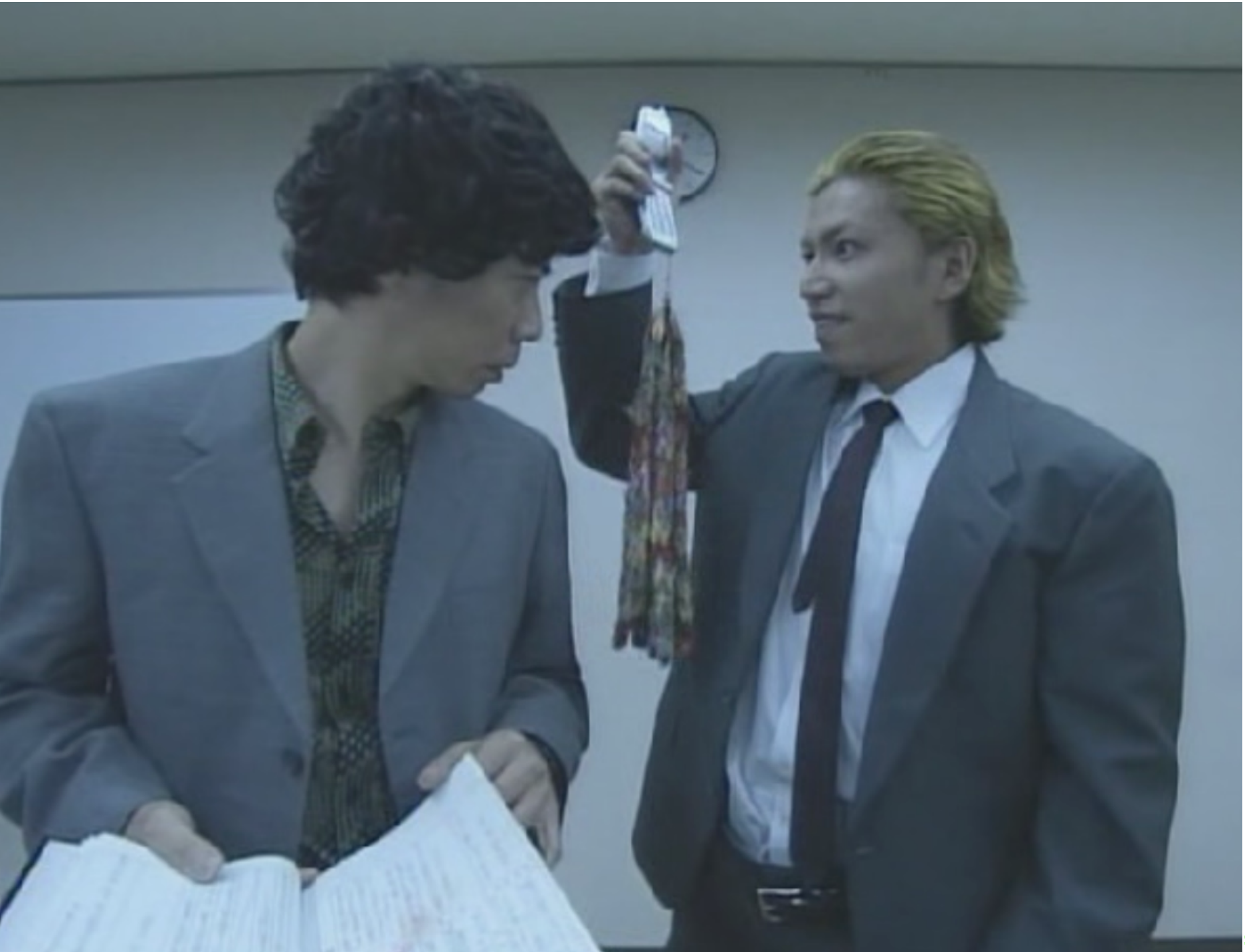
Course Home Page

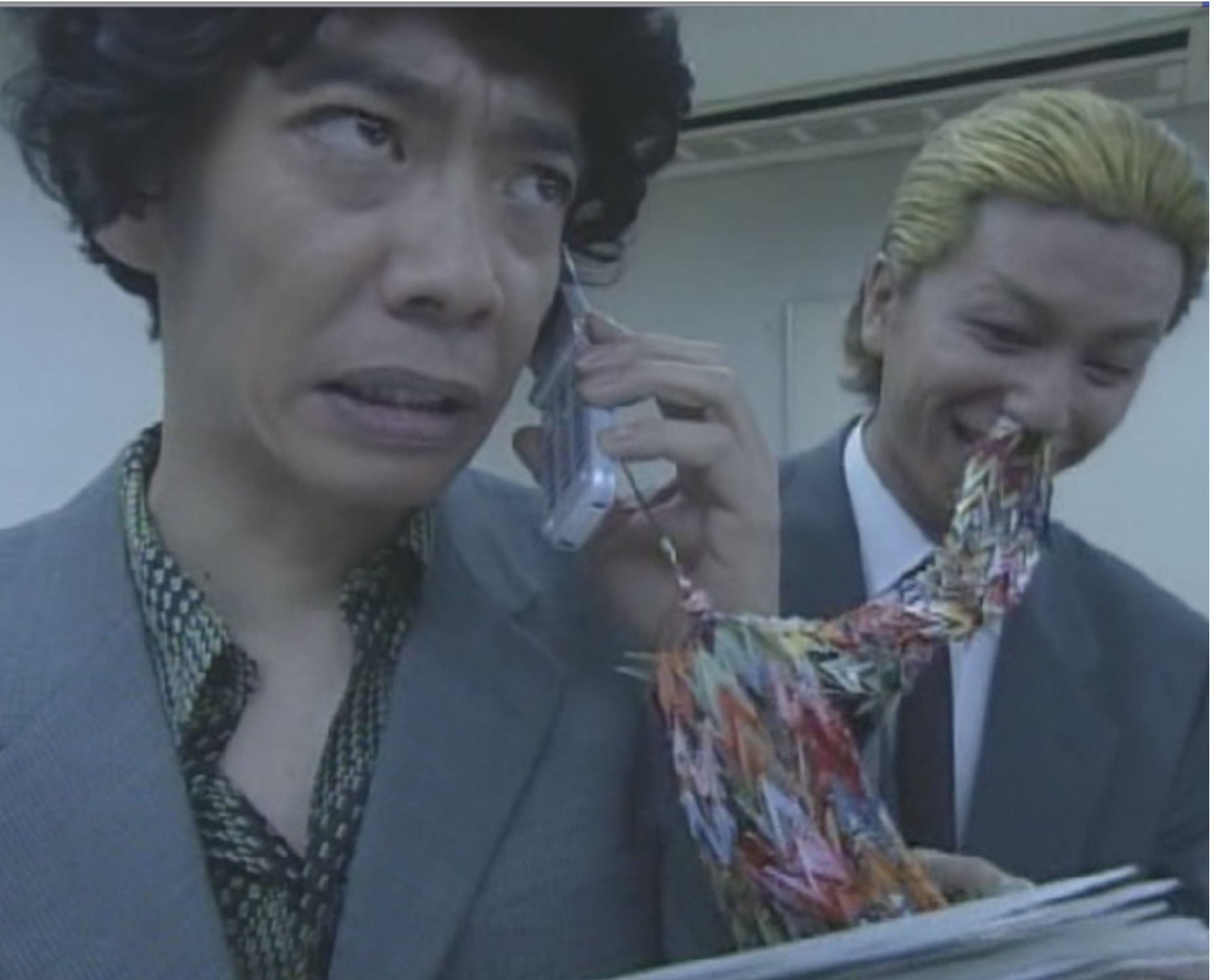
- **google: virginia cs 4610**
- **http://www.cs.virginia.edu/~weimer/4610**
- Lectures slides are available before class
  - You should still take notes!
- Assignments are listed
  - also grading breakdown, regrade policies, etc.

- Use the class Collab forum for all public questions

Discussion Sections

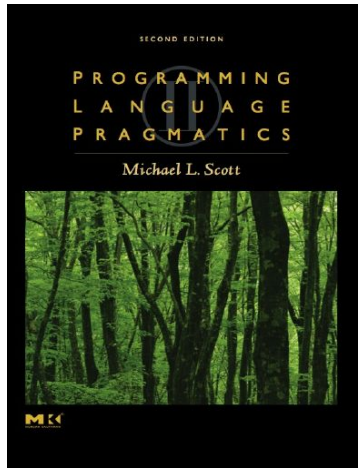
- There will be one sixty-minute "structured office hour" each week
  - Hosted by Zak Fry, the TA
- We will not take attendance, but you are encouraged to show up each week
  - Notes posted on web
  - For your benefit!
- Answer questions, go over lecture material, help and hints on the homework and projects
- Next Week: pass around time signup sheet





Course Structure

- Course has theoretical and practical aspects
  - Best of both worlds!
- Need both in programming languages!
- Reading = both
  - Many external and optional readings
- Written assignments = theory
  - Class hand-in, right before lecture, 0-5 points
- Programming assignments = practice
  - Electronic hand-in
- Strict deadlines
  - Ask me why ...

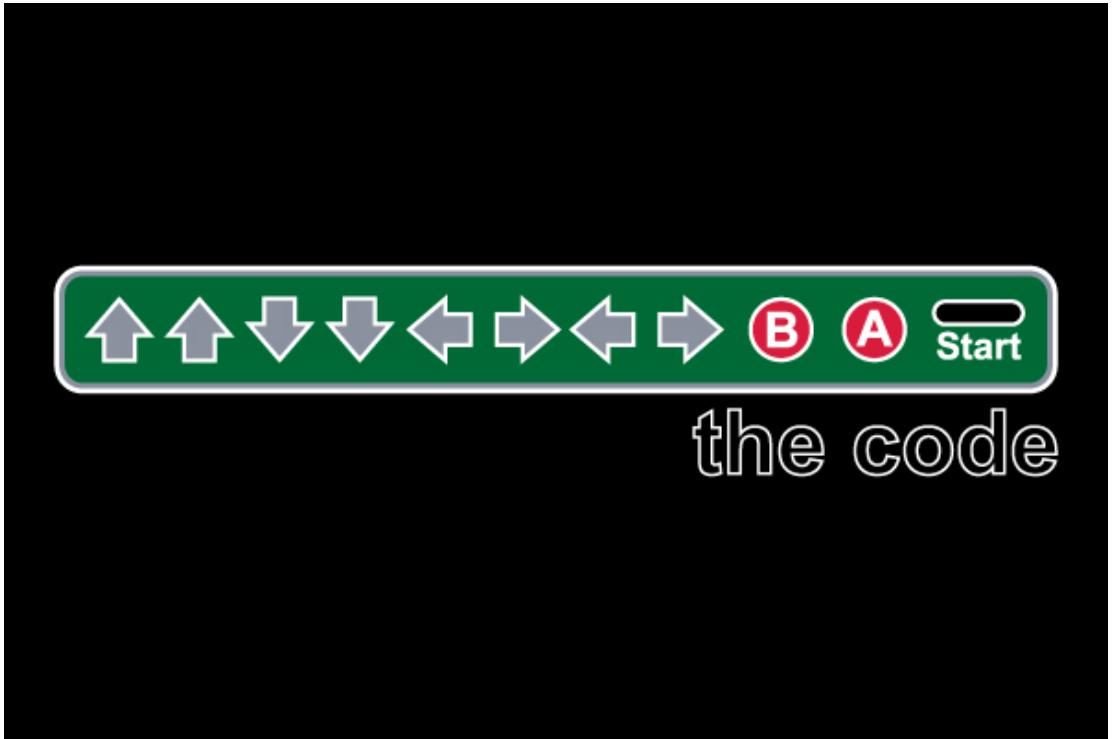


Academic Honesty

- Don’t use work from uncited sources
  - Including old code
- We often use plagiarism detection software

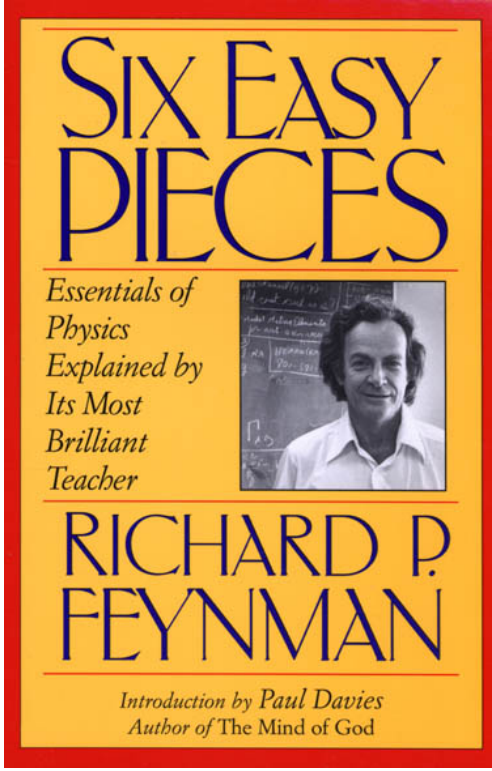
Academic Honesty

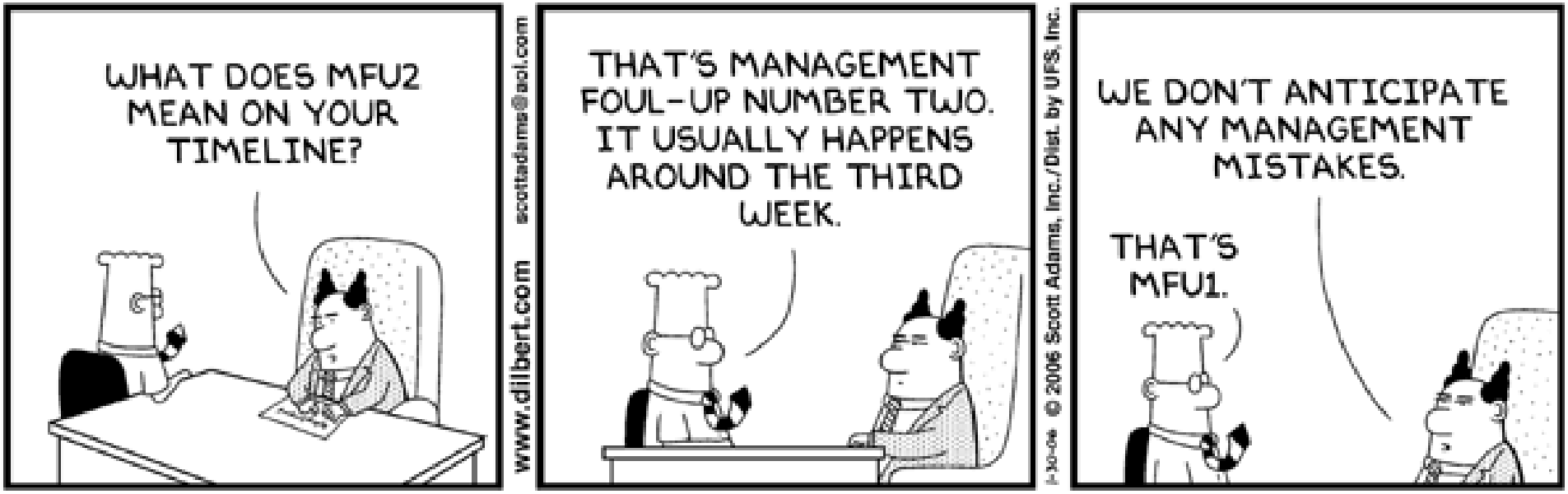
- Don't use work from uncited sources
  - Including old code
- We often use plagiarism detection software



The Course Project

- A big project: a Compiler!
- … in five (plus one) easy parts
- Start early!





© Scott Adams, Inc./Dist. by UFS, Inc.

Course Goals

- At the end of this course, you will be acquainted with the fundamental concepts in the **design** and **implementation** of high-level programming **languages**. In particular, you will understand the **theory** and **practice** of **lexing**, **parsing**, **semantic** analysis, and **code** generation. You will also have gained practical experience programming in multiple **different** languages.

Credit Where Credit Is Due

- “Extremely unreasonable amount of programming efforts expected ...”
- “I don't know what you should do, but this class had ruined my life ...”
- “The course is very intense and demanding ...”
- “A ridiculous amount of work but that was known information from the first day of the course ...”
- “This course was the hardest course I have taken thus far ...”

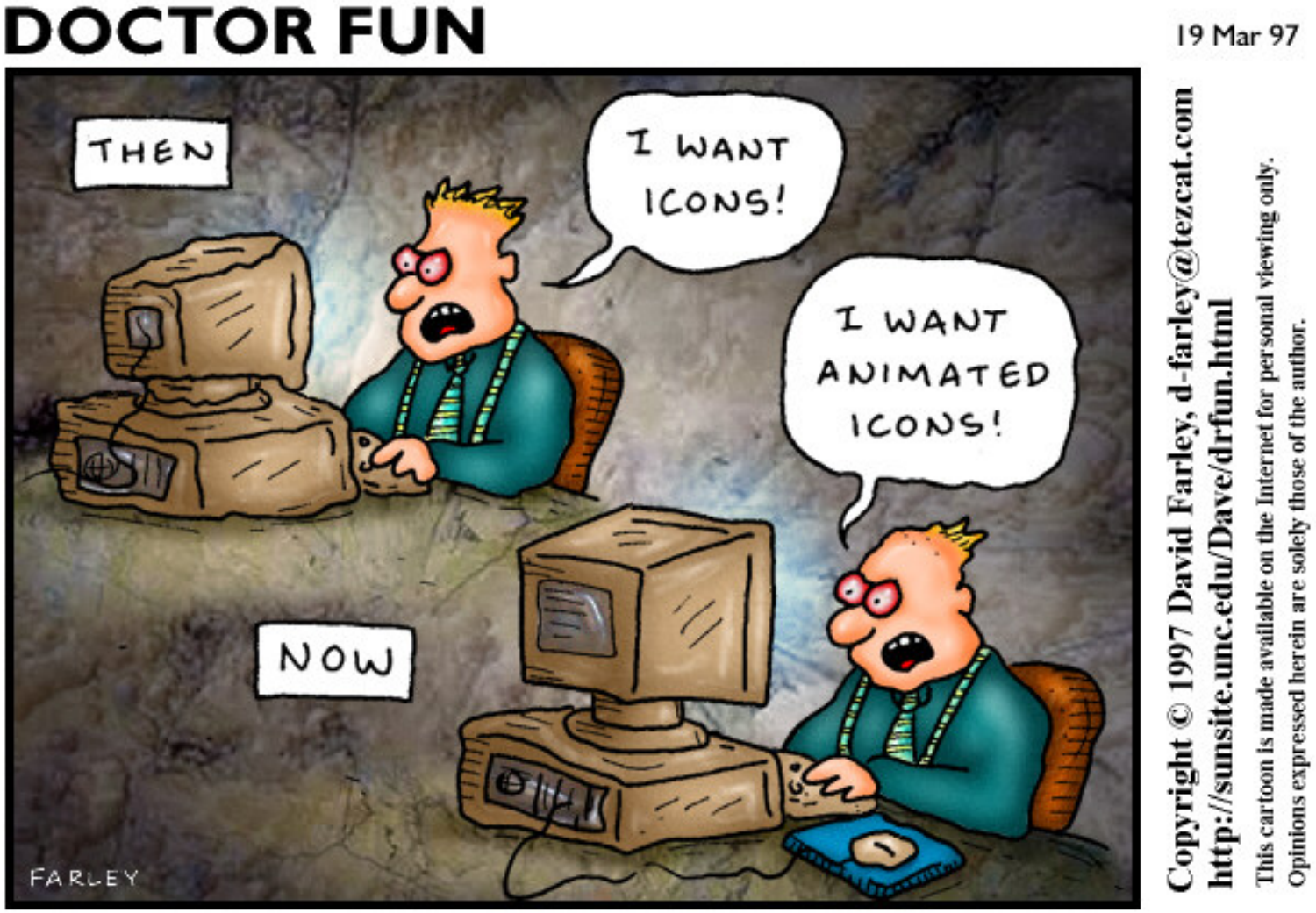
Five Credits

- Those comments were for last year's version of this class, which was worth three credits.
- This year, CS 4610 can effectively be worth **five credits**.
  - We're giving you the credit you deserve for the work you'll accomplish.
- Sign up for two credits of CS 4993-04 (37675)
  - “Independent Study”, letter grade
- You'll get the same grade in both places.

How are Languages Implemented?

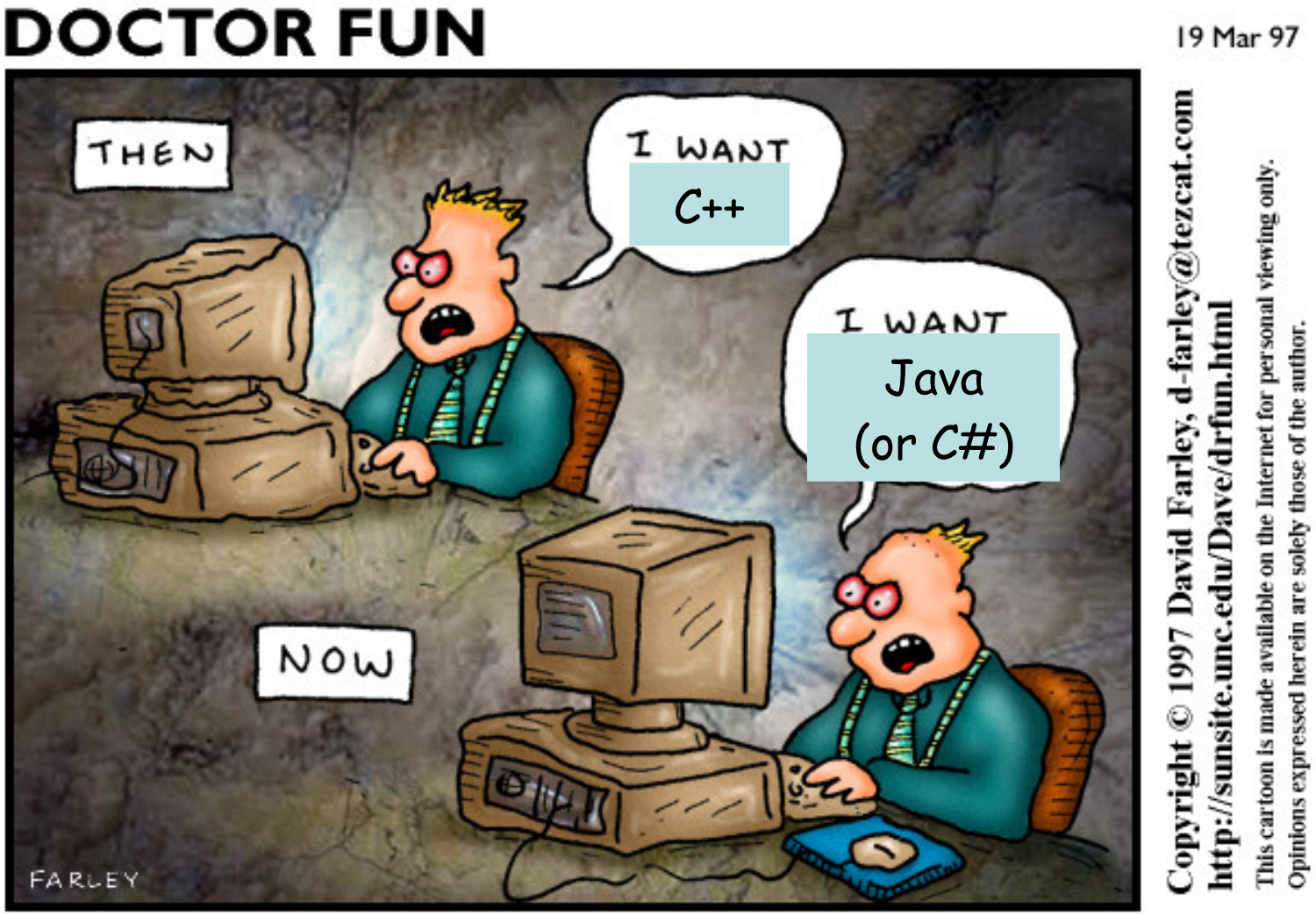
- Two major strategies:
  - Interpreters (take source code and run it)
  - Compilers (translate source code, run result)
  - Distinctions blurring (e.g., just-in-time compiler)
- Interpreters run programs “as is”
  - Little or no preprocessing
- Compilers do extensive preprocessing
  - Most implementations use compilers

Don’t We Already Have Compilers?



Progress

Dismal View Of Prog Languages



(Short) History of High-Level Languages

- 1953 IBM develops the 701 "Defense Calculator"
  - 1952, US formally ends occupation of Japan
  - 1954, Brown v. Board of Education of Topeka, Kansas
- **All programming done in assembly**
- **Problem: Software costs exceeded hardware costs!**
- John Backus: "Speedcoding"
  - An interpreter
  - Ran 10-20 times slower than hand-written assembly

FORTRAN I

- 1954 IBM develops the 704
- John Backus
  - Idea: translate high-level code to assembly
  - Many thought this impossible

- 1954-7 FORTRAN I project
- By 1958, >50% of all software is in FORTRAN
- Cut development time dramatically
  - (2 weeks → 2 hours)

FORTRAN I

- The first **compiler**
  - Produced code almost as good as hand-written
  - Huge impact on computer science
- Led to an enormous body of theoretical work
- Modern compilers keep the outlines of FORTRAN I

© Scott Adams, Inc./Dist. by UFS, Inc.

Q: TV (100 / 842)

- In this 1985-1992 ABC television series, the gunless title character Angus works for Pete and the Phoenix Foundation and makes heavy use of his Swiss Army knife and duct tape.

Bulwer-Lytton vs. World

- Morgan said his goal is not to condemn manufacturers with blanket statements, but to correct the snag. "The idea that quality is based on thread count is not some old yarn – it's woven into the fabric of our society," Morgan said. "But the system for quality control is threadbare. It's coming apart at the seams. We can't pull the covers over our heads and ignore it any longer."
- Those two were like a pair of maracas, loud and obnoxious when they were together.
- Dolores breezed along the surface of her life like a flat stone forever skipping across smooth water, rippling reality sporadically but oblivious to it consistently, until she finally lost momentum, sank, due to an overdose of fluoride as a child which caused her to lie forever on the floor of her life as useless as an appendix and as lonely as a five-hundred-pound barbell in a steroid-free fitness center.
- Part of her knew she shouldn't tease him, actually, all of her knew she shouldn't tease him ... it was like playing hacky sack with a grenade.
- The city had faded out of Chandler over the years, like the color had faded out of his clothes ... he washed everything until it fell apart, quintessential bachelor, who still treated the washing machine like a squat, boxy monster that only gave his clothes back because it was biding its time to attack.

The Structure of a Compiler

- Lexical Analysis
- Parsing
- Semantic Analysis
- Optimization (optional)
- Generate Machine Code
- Run that Machine Code

This is the class programming project!

The first 3, at least, can be understood by analogy to how humans comprehend English.

Lexical Analysis

- First step: recognize words.
  - Smallest unit above letters

This is a sentence.

- Note the
  - Capital "T" (start of sentence symbol)
  - Blank " " (word separator)
  - Period "." (end of sentence symbol)

More Lexical Analysis

- Lexical analysis is not trivial. Consider:

  How d'you break "this" up?

- Plus, programming languages are typically more cryptic than English:

  ***p->f += -.12345e-6**

And More Lexical Analysis

- Lexical analyzer divides program text into "words" or tokens

  if x == y then z = 1; else z = 2;

- Broken up:

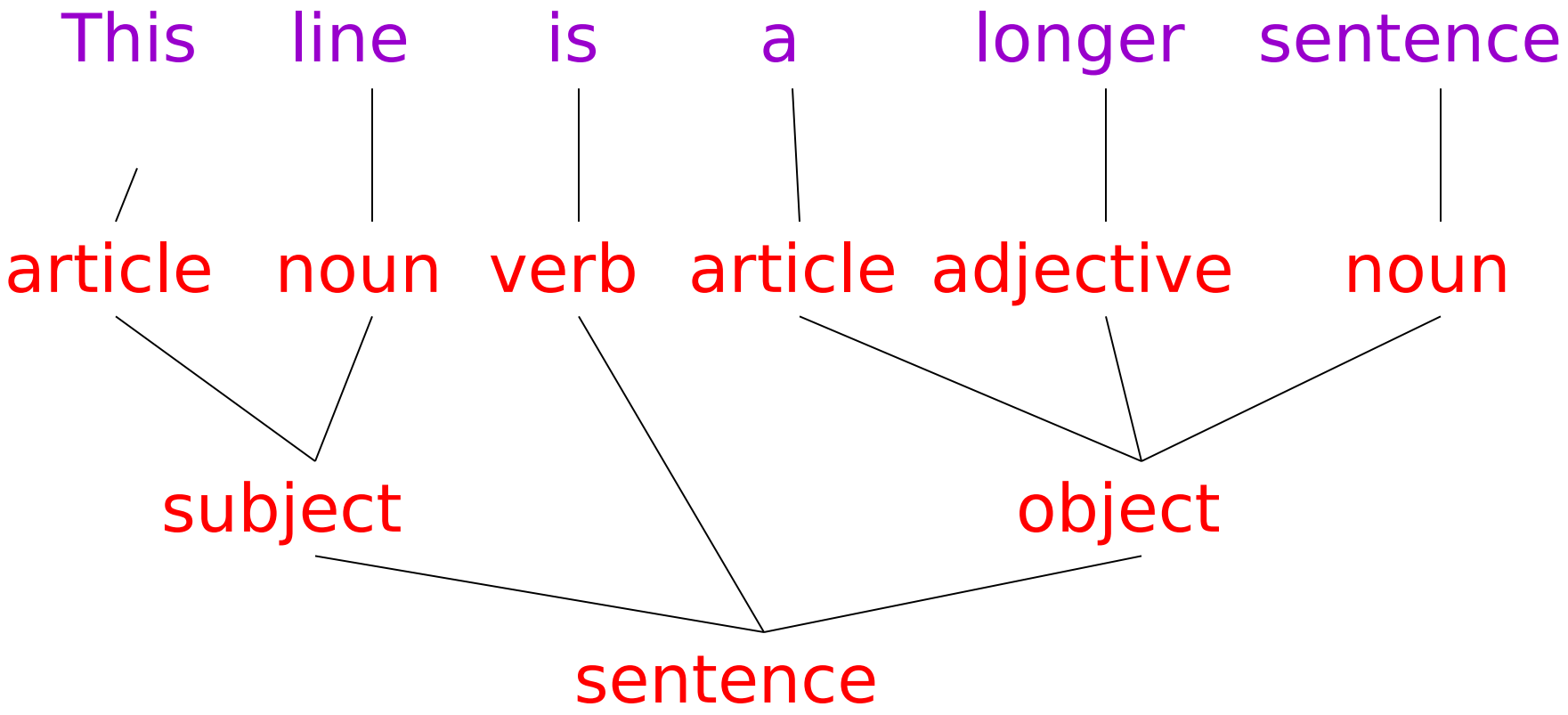
  if, x, ==, y, then, z, =, 1, ;, else, z, =, 2, ;

Parsing

- Once words are understood, the next step is to understand sentence structure

- Parsing = Diagramming Sentences
  - The diagram is a tree

Diagramming a Sentence

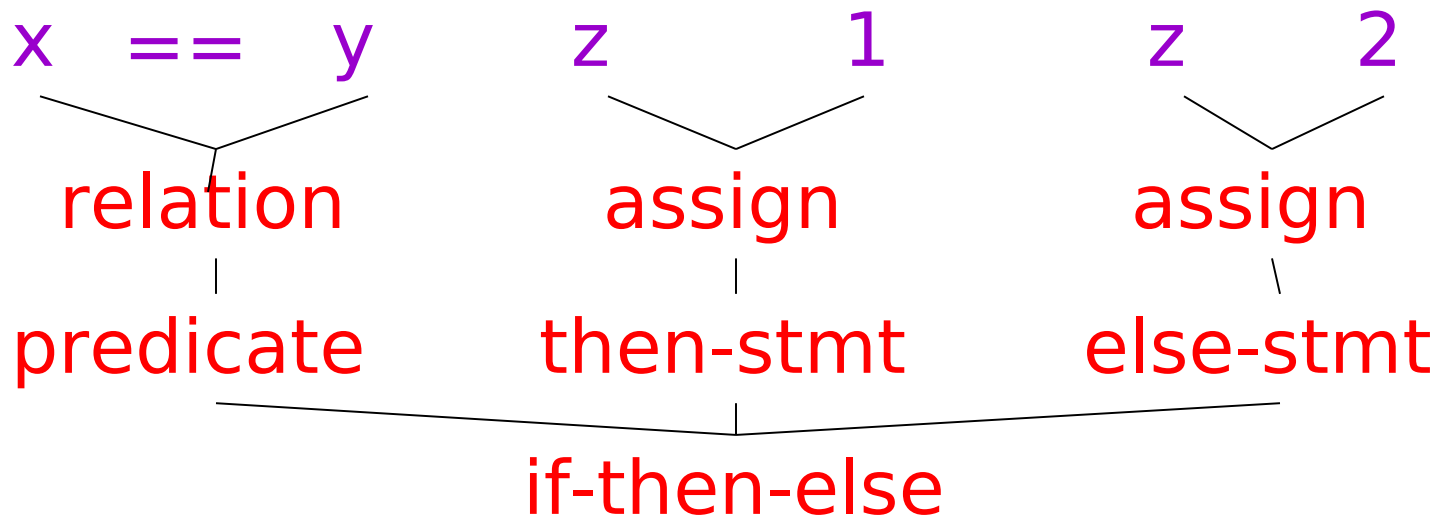


Parsing Programs

- Parsing program expressions is the same
- Consider:

  if x == y then z = 1; else z = 2;

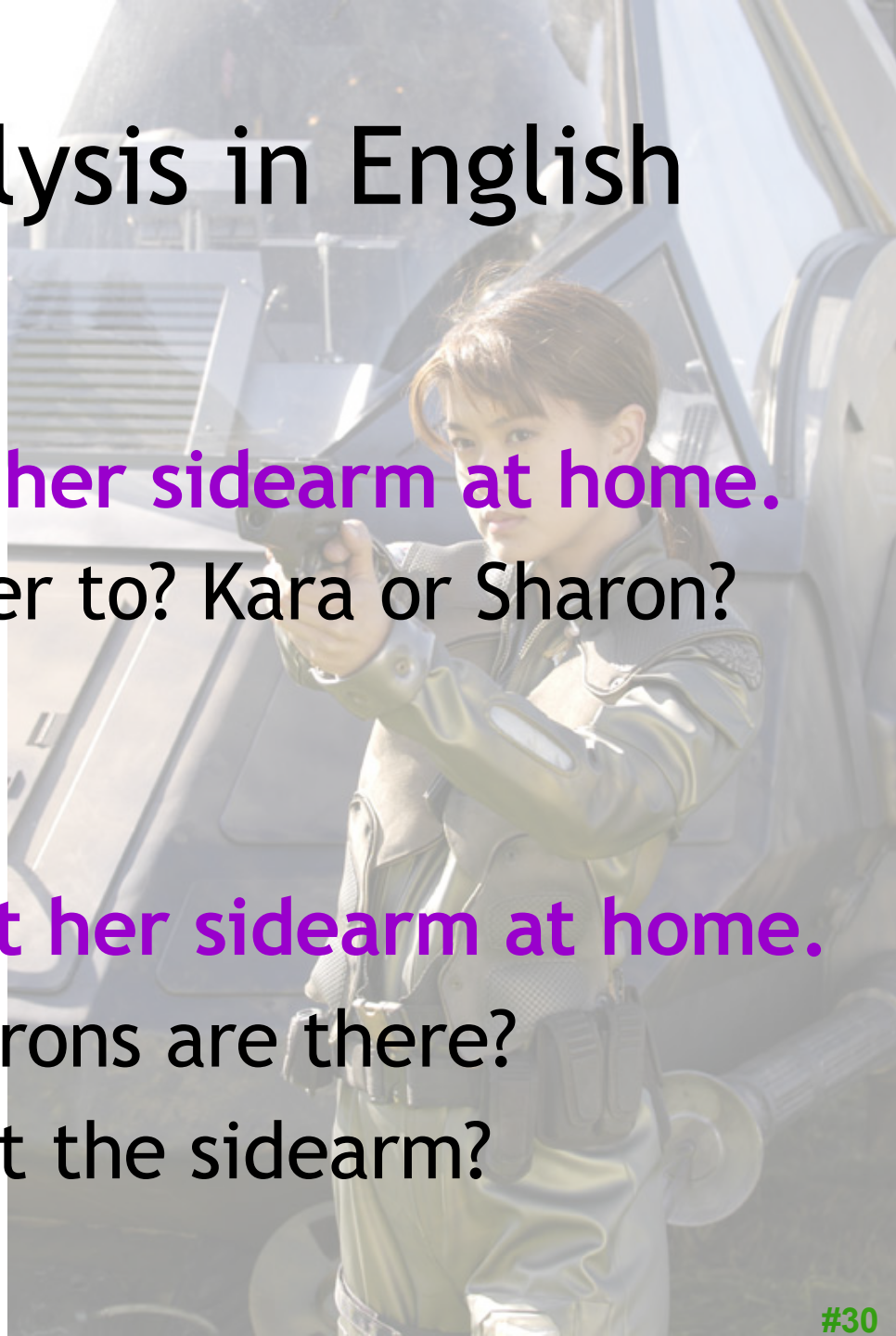
- Diagrammed:



Semantic Analysis

- Once sentence structure is understood, we can try to understand "meaning"
  - But meaning is too hard for compilers
- Compilers perform limited analysis to catch inconsistencies: reject bad programs early!
- Some do more analysis to improve the performance of the program

Semantic Analysis in English



- Example:

  Kara said Sharon left her sidearm at home.

  What does “her” refer to? Kara or Sharon?

- Even worse:

Sharon said Sharon left her sidearm at home.

How many Sharons are there?

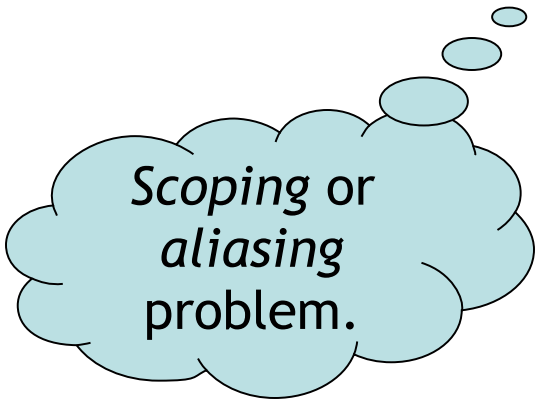
Which one left the sidearm?

It's context-sensitive!

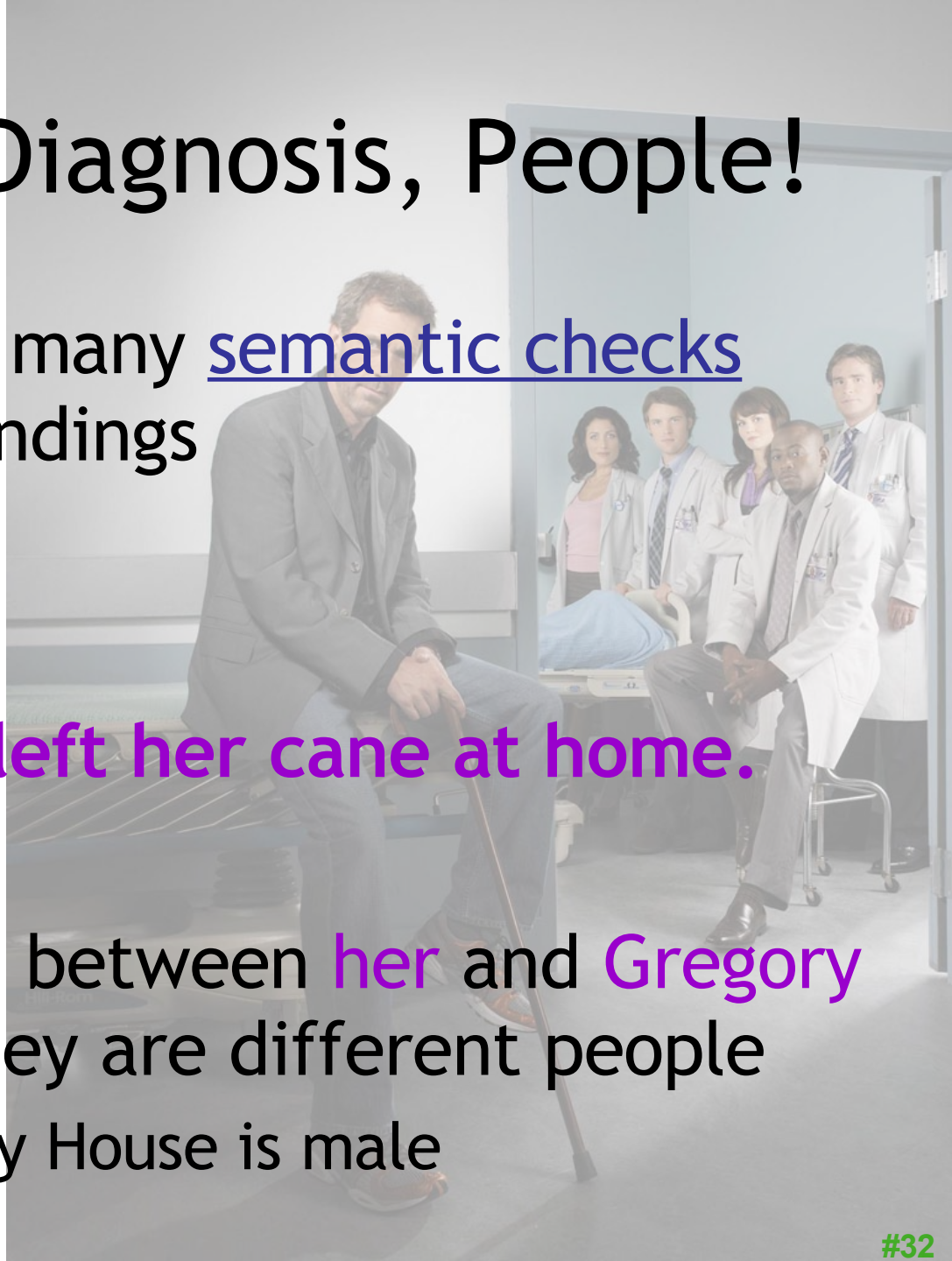
Semantic Analysis in Programming

- Programming languages define strict rules to avoid such ambiguities
- This C++ code prints "4"; the inner definition is used

```
{
  int Sydney = 3;
  {
    int Sydney = 4;
    cout << Sydney;
  }
}
```



Differential Diagnosis, People!



- Compilers perform many semantic checks besides variable bindings

- Example:

  Gregory House left her cane at home.

- A "type mismatch" between her and Gregory House; we know they are different people
  - Presumably Gregory House is male

Optimization

- No strong counterpart in English, but akin to editing

- Automatically modify programs so that they
  - Run faster
  - Use less memory
  - In general, conserve some resource

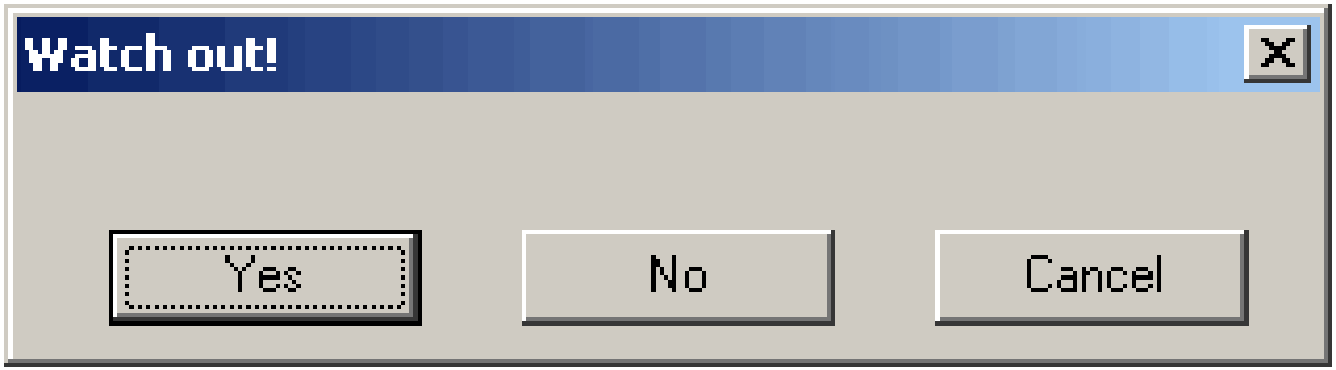
- The project **extra credit** optimization

Code Generation

- Produces assembly code (usually)
  - which is then assembled into executables by an assembler
- A translation into another language
  - Analogous to human translation
- This class: produce simple machine code
  - A cross between Java bytecode, x86, and RISC

Issues

- Compiling and interpreting are almost this simple, but there are many pitfalls.
- Example: How are bad programs handled?
- Language design has big impact on compiler
  - Determines what is easy and hard to compile
  - Course theme: **trade-offs in language design**



Languages Today

- The overall structure of almost every compiler & interpreter follows our outline

- The proportions have changed since FORTRAN
  - Early: lexing, parsing most complex, expensive

  - Today: optimization dominates all other phases, lexing and parsing are cheap
  - ... but still matter, ramble ramble ...

Trends in Languages

- Optimization for speed is less interesting. But:
  - scientific programs
  - advanced processors (Digital Signal Processors, advanced speculative architectures)
  - Small devices where speed = longer battery life
- Ideas we'll discuss are used for improving code reliability:
  - memory safety
  - detecting concurrency errors (data races)
  - type safety
  - automatic memory management
  - …

Why Study Prog. Languages?

- Increase capacity of expression
  - See what is possible
- Improve understanding of program behavior
  - Know how things work “under the hood”
- Increase ability to learn new languages
- Learn to build a large and reliable system
- See many basic CS concepts at work

What Will You Do In This Class?

- Reading (textbook, outside sources)
- Learn about different kinds of languages
  - Imperative vs. Functional vs. Object-Oriented
  - Static typing vs. Dynamic typing
  - etc.
- Learn to program in different languages
  - Python, Ruby, ML, “Cool” (= micro-Java)
- Complete homework assignments
- Write an interpreter!

What Is This?

A lungo il mio cuore di tali ricordi ha voluto colmarsi!

Come un vaso in cui le rose sono state dissetate:

Puoi romperlo, puoi distruggere il vaso se lo vuoi,

Ma il porfumo delle rose sarà sempre tutt'intorno.

Długo, długo moje serce przepełnione było takimi wspomnieniami!

Były jak waza, w której kiedyś róże destylowały:

Możesz sprawić by pekła, możesz gruchotać waze jeśli chcesz,

Ale zapach róż bedzie wciaż czuć dookoła.

Mon coeur est brûlant rempli de tels souvenirs

Comme un vase dans lequel des roses ont été distillées:

Tu peux le briser, tu peux détruire le vase si tu le désires,

Mais la senteur des roses sera toujours là.

Lang, lang soll die Erinnerung in meinem Herzen klingen!

Gleich einer Vase, drin Rosen sich einst tränkten:

Lass sie zerbrechen, lass sie zerspringen,

Der Duft der Rose bleibt immer hängen.

Muito, muito tempo seja meu coração preenchido com tais lembranças!

Tal qual o vaso onde rosas foram uma vez destiladas:

Pode quebrar, pode estilhaçar o vaso se o desejas,

Mas perdurará para sempre o aroma das rosas perfumadas.

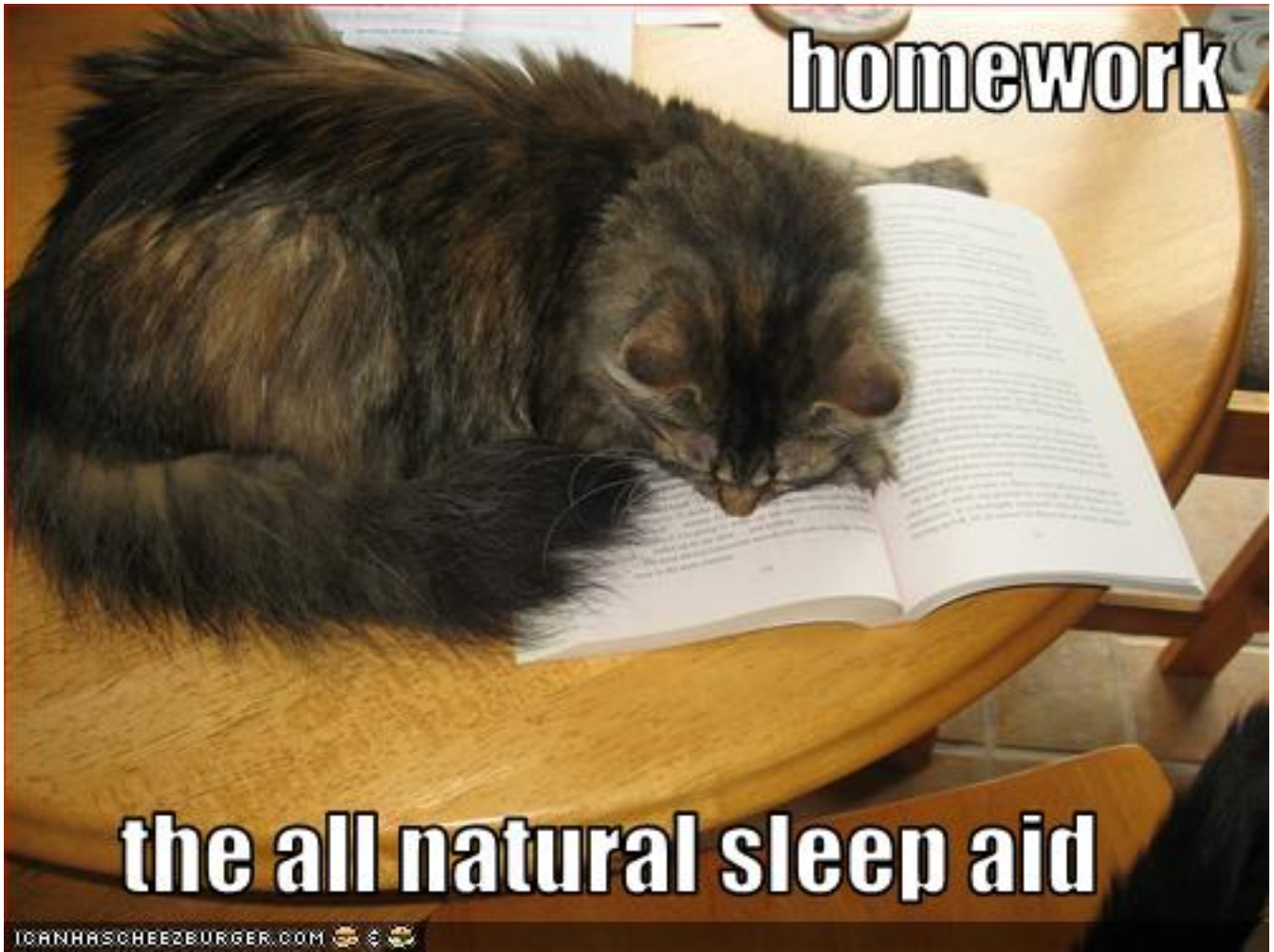
The Rosetta Stone

- The first programming assignment involves writing the same simple (50-75 line) program in:
  - **Ruby, Python, OCaml, Cool and C**
- PA0, **due Wed Sep 02**, requires you to write the program in two languages (you pick)
- PA1, due one week later, requires all five

Long, long be my heart with such memories fill'd!
Like the vase in which roses have once been distill'd:
You may break, you may shatter the vase if you will,
But the scent of the roses will hang round it still.
- Thomas Moore (Irish poet, 1779-1852)

Start The Homework Now

- It may help you decide whether to stay in this course



Homework

- Scott Book, parts of Chapter 10 (for Thursday)
- Get started on PA0 (due in 8 days)

Questions?