CS 100

Tuesday

22 September 2015

Today’s Agenda

0. Mr. Jim Schroder: UPS
1. Announcements
2. Resumé Return
3. Selfie review
4. Teem Geek Intro: “The Genius Programmer”
5. What is Computer Science? (Part II: CS Ideas)

1. Announcements

- Prep for next week: reading assignment internships)
- Clickers: click-in is now *obligatory* for attendance
- Calculus exams [today is first one: slip out at 4:50pm]

2. Resumé Return
(please be patient, and quiet)

3. Selfies

Discussion of First Assignment

4. Team Geek

(Launch clickers)

5. What is Computer Science? Part II: CS Ideas

Computer Science Joke

Q: Why do programmers always mix up Halloween and Christmas?

A: Because Oct 31 == Dec 25!

Computer Science Joke

A man flying in a hot air balloon suddenly realizes he’s lost. He reduces height and spots a woman down below. He lowers the balloon further and shouts to get directions, "Excuse me, can you tell me where I am?"

The woman below says: "Yes. You're in a hot air balloon, hovering 30 feet above this field."

"You must work **in Information Technology**," says the balloonist.

"I do" replies the woman. "How did you know?"

"Well," says the balloonist, "everything you have told me is technically correct, but It's of no use to anyone.”

Computer Science Joke

The woman below replies, "You must work in management."

"I do," replies the balloonist, "But how'd you know?"*

"Well", says the woman, "you don’t know where you are or where you’re going, but you expect me to be able to help. You’re in the same position you were before we met, but now somehow it’s my fault."

Computing Science Ideas

- Paradigm, n: EXAMPLE; PATTERN, esp. an outstandingly clear or typical example or archetype...
- In Computing Science (as in many areas of life), certain ideas/techniques appear over and over
- The successful computing professional recognizes these concepts and knows how to apply them in a variety of contexts

CS Big Ideas

- Hierarchical Abstraction
- Compositionality
- Caching

Hierarchical Abstraction

- Basic idea: organize and name things in hierarchical groups
  - Refer to many items by one name
    - What is being abstracted?
  - Examples
    - Citizens of the United States/Kentucky/Madison County/Richmond
    - Students in the University of Kentucky/College of Engineering/Computer Science Department/Lab for Advanced Networking

Hierarchical Abstraction

- Especially useful for, e.g., routing

  Ms. Sue Jackson

  123 Main Street

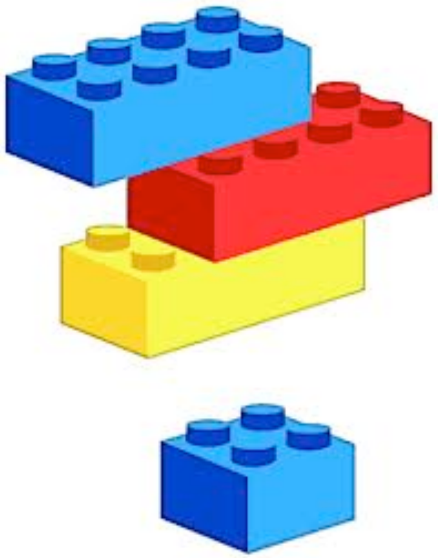
  Boston, MA 02215-0139

Geometry / Geo Example

- World map: divided into regions
  - Regions divided into sub-components
  - Zoom level excludes most data except for the region of interest
- Street view data
  - Viewpoint location / direction limits data required to generate view
  - Possible movements allow “pre-fetching”

Compositionality

- Lots of systems in the world work like this:
  - A few small, elementary things are "given"
  - Some way is defined to compose things to get new things
- Canonical example:
- Others?



Peano's Axioms

- 0 is a natural number.
- "=" denotes equality.
  - reflexive, transitive, symmetric
- If m is a natural number, then S(m) is a natural number (called the "successor" of m).
- For any natural number m, $S(m) \neq 0$.
- For any natural numbers m and n,

  if $S(m) = S(n)$, then $m = n$
- For any set P of natural numbers, if
  - P contains 0
  - For any natural number m, if P contains m, then P contains S(m)

  then P contains all natural numbers

Peano's Axioms, cont.

- Addition:
  - For any natural numbers a, b: we can define $a + b$ as follows:
    - $a + 0 = a$
    - $a + S(b) = S(a + b)$
- Multiplication:
  - For any natural numbers a, b: we can define $a \times b$ as follows:
    - $a \times 0 = 0$
    - $a \times S(b) = (a \times b) + a$

Peano's Axioms, cont.

- Can show that addition
  - is commutative
  - is associative
- Can show that multiplication
  - is commutative
  - is associative
  - distributes over addition

Composition and Programming

- Many programming languages have this structure:
  - A few "atomic" statements:
    - Assignment: x = 3;
    - Empty:
  - Composition operators make composite statements:
    - Sequencing: x = 3; y = x + 1;
    - Conditional: if (A[i] == target) j = i; else i = i+1;
    - Iterative: while (A[i] != target) i = i+1;
    - Functional abstraction: Single expression represents a whole computation

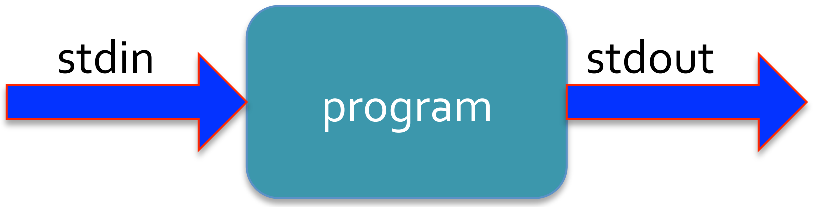
Composition and Programming

- The idea: build up large programs/systems by composing smaller ones
  - Functional (or procedural) abstraction is the primary scaling mechanism* in programming

    *i.e., a way to may things work in big systems

Composing Programs: The Unix model

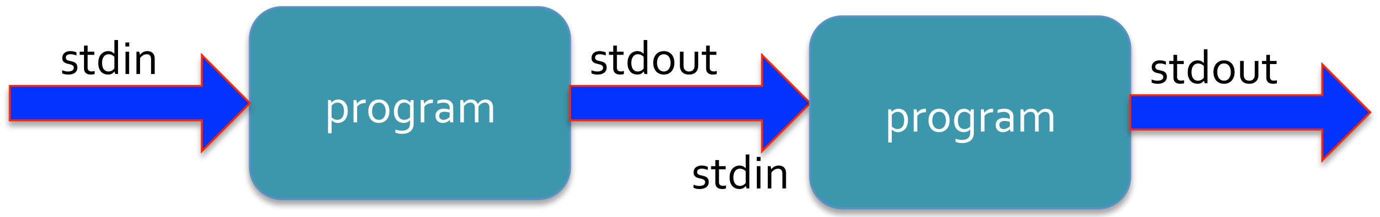
- Unix Systems convention (dates to 1970's):
  - Programs communicate via [standard] input and output streams:
    - Read data from standard input
    - Write results to standard output



  - Small programs that operate on text input
    - Example: grep

Composing Programs: The Unix model

- Composing programs:
  - "Pipeline" = Connect stdout of one program to stdin of another



  - Example: sort lines in a text file containing "@":
    - grep '@' foo.txt | sort

Composition operator

The Power of the Unix Model

- You have a file (named “roster.txt”), containing many lines, each with:
  - Name
  - Student ID
  - Username
  - Email
  - Degree, Major-Minor, Classification, Hour(s), Status

  Example:

```
"Seales, William Brent" 10210288 CZQI1442  seales@uky.edu  BSCS-EN
PCOS-BSCS  Undergrad - Freshman    1       Enrolled
```

The Power of the Unix Model

- You are asked to extract just the list of email addresses, and present them in sorted order

  How will you do it?

The Power of the Unix Model

- You are asked to extract just the list of email addresses

  Windows solution:.... ?

  Unix solution: run the pipeline:

```
cut -f 4 roster.txt  | more
cut -f 4 roster.txt | grep -i david
cut –f 4 roster.txt  |  fgrep  ‘@’  |  sort
cut -f 4 roster.txt | fgrep '@' | sort | more
cut -f 4 roster.txt | fgrep '@' | sort >&
  tmp.txt
```

Unix and Windows

- Unix model:
  - Data (files) exist independent of applications, may be processed by many different programs
  - Interaction with the user is optional
  - Build small, simple (correct!) programs and compose them
- Windows model (now also the smartphone model)
  - Every data file has an (unique?!) associated program that is the way to access it
    - Each program owns its own data
  - Every program is interactive
  - No composition of programs ☹

Caching

- Idea: once you have expended some resources to obtain some information, you may want to store it in case you need it again

  (at least for a little while)
- Non-computing examples?
  - atom-based examples are rare
- Computing example(s)?

Modern Wonder: the Domain Name System

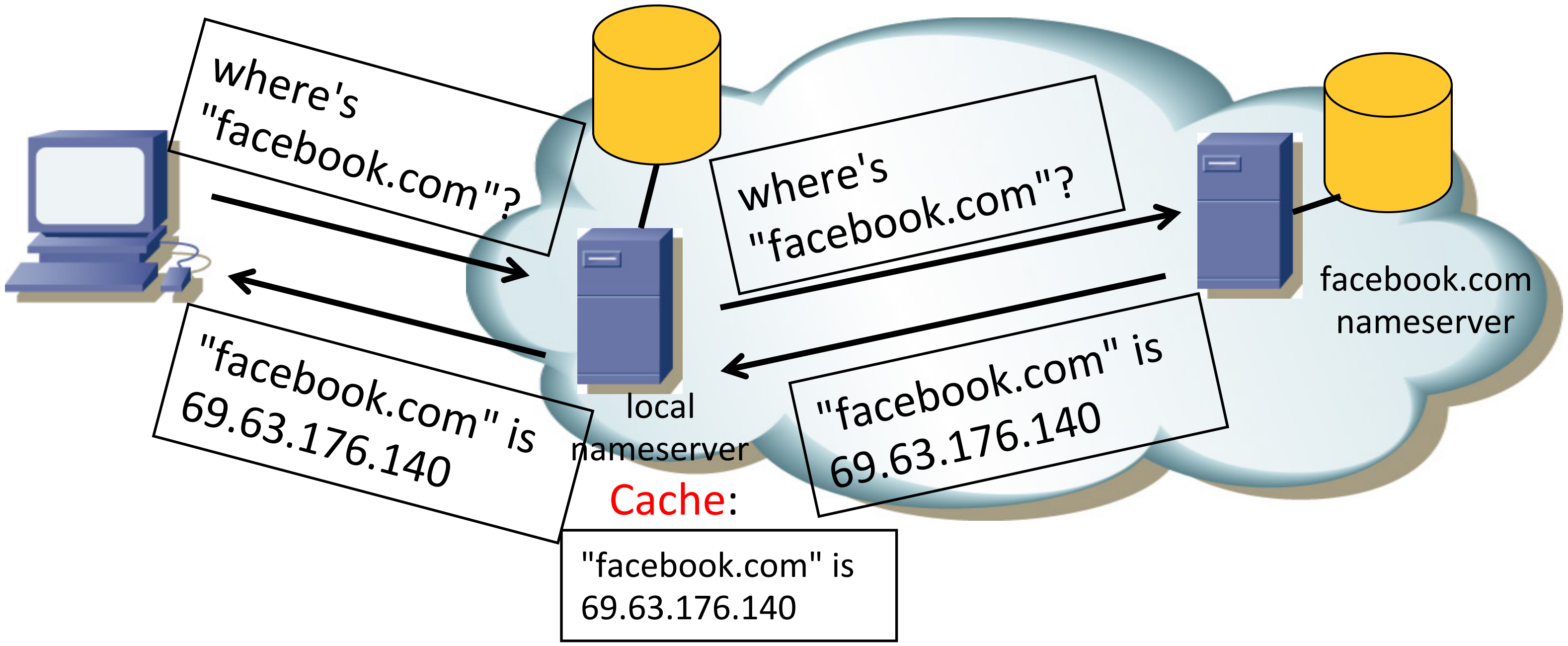
- Internet protocols deal with 32-bit IP addresses
  - 10000000101000111000110011011101, or 0x80A38CDD, or 128.163.140.221
- How packets are routed based on IP addresses...
- Names are easier for humans
- Problem: design a system for resolving names to (Internet) numbers and other information

  Require the following characteristics:

  - Scalable to billions of name-number pairs
  - Distributed control: UK gets to decide what names end in "uky.edu"; CS dept gets to decide what names are under "cs.uky.edu" (but not pa.uky.edu or delta.com or...)
  - Robust: no single point of failure

Domain Name System

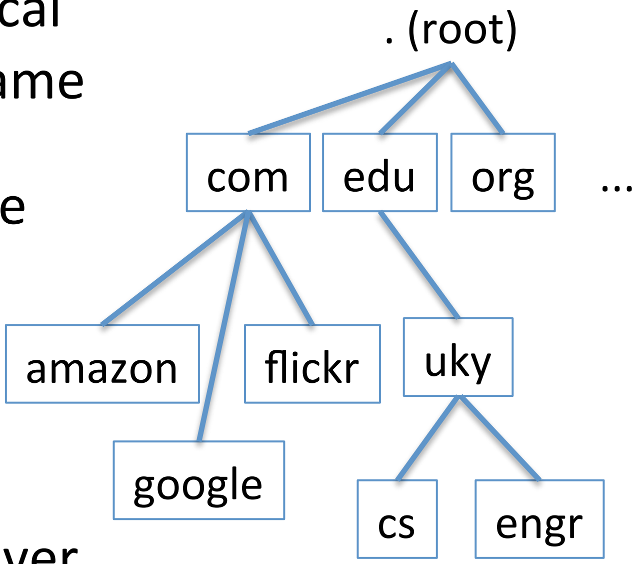
- Every URL contains a DNS name
  - http://video.google.com
  - http://protocols.netlab.uky.edu/~calvert/
- Every time you click, your computer sends a message to its local DNS server asking to resolve that DNS name to an IP address
- This happens at least millions of times per second worldwide


where's "facebook.com"?
"facebook.com" is 69.63.176.140
local nameserver
Cache:
"facebook.com" is 69.63.176.140
where's "facebook.com"?
"facebook.com" is 69.63.176.140
facebook.com nameserver

Scaling the DNS

The DNS depends on two standard CS techniques:

- Hierarchical Abstraction
  - The namespace authority is hierarchical
  - Names in the same group have the same suffix (e.g., ".org.")
  - Each group only concerned with those below it
- Caching
  - Remember what you've learned
  - Exploit locality of reference
  - Amortize the cost of one resolution over many queries



Take Aways

Crucial Computing Science concepts:

- Hierarchy
  - Subdivide to manage scale
- Composition
  - Fundamental idea for abstraction engineering
- Caching
  - Key performance aid