CMPT 140: Introduction to Programming

9 Sep 2010
CMPT140
Dr. Sean Ho
Trinity Western University

- *Pick up syllabus*
- *Sign-in on sheet*

TRINITY
WESTERN
UNIVERSITY

Outline for today

- **Course information**
  - Course website, syllabus, schedule
- Guest alumnus: Kevin Tan / Accenture
- Programming as problem-solving
  - Tools, toolsmiths, toolboxes
  - Top-down vs. bottom-up design
  - Example: woodcutting
- Demo of our Python programming environment
- Tour of the CSI computing lab
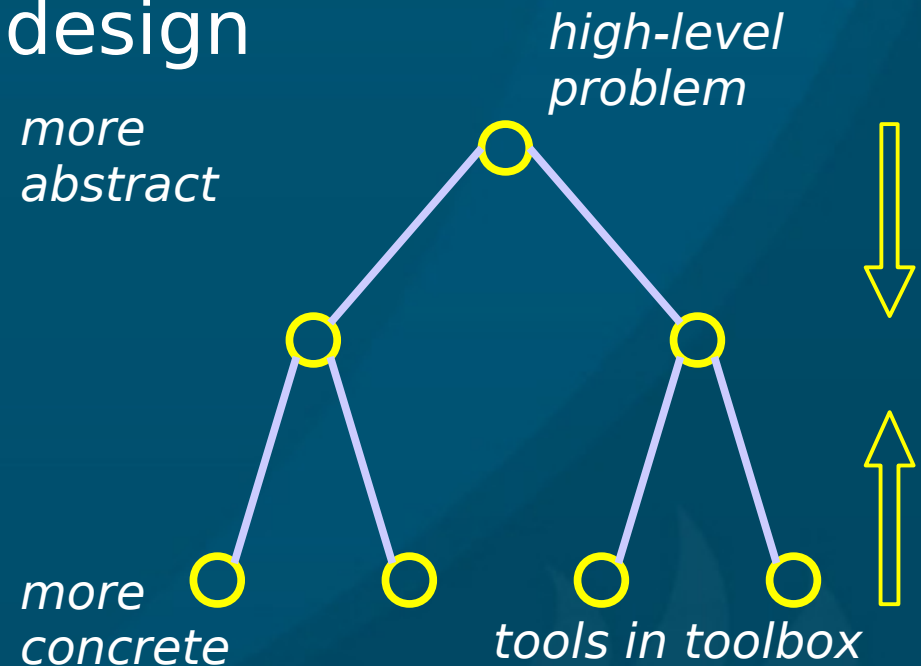
About CMPT 140

- Software development is about problem-solving

- Computing science is interdisciplinary:

  - Users, clients: must interact with and understand people!

- This course is not about "cracking" (trying to destroy or break into things)

- It's not even about "hacking" (lone wizard staring at computer late at night)

- It's about solving problems using software!

Programming is problem solving

- What are problems we might want to solve?
  - One is at the bottom of Mt. Baker and wishes to be at the top
  - A large mailing list needs to be sorted alphabetically to search for duplicates
  - A business owner needs to maximize customer satisfaction and profits
  - Two children are listening to the Canucks hockey game but ought to be sleeping
- The right tool for the right job
  - Computers are not always the right tool!

TRINITY
WESTERN
UNIVERSITY

Problem solving

- Top-down vs. bottom-up design

- **_W_**rite everything down
- **_A_**pprehend the problem
- **_D_**esign a solution
- **_E_**xecute the plan
- **_S_**crutinize the results

high-level problem

more abstract

more concrete

tools in toolbox

TRINITY WESTERN UNIVERSITY

The Art of the Toolsmith

- Computers and software are tools; Computing scientists are toolsmiths

- The success of the tool is evaluated by the user, not by the toolmaker!

```
+    threadfn = create->threadfn;
+    data = create->data;
+
+    /* Block and flush all signals (in case we're not from keventd). */
+    sigfillset(&blocked);
+    sigprocmask(SIG_BLOCK, &blocked, NULL);
+    flush_signals(current);
+
+    /* By default we can run anywhere, unlike keventd. */
+    set_cpus_allowed(current, mask);
+
+    /* OK, tell user we're spawned, wait for stop or wakeup */
+    __set_current_state(TASK_INTERRUPTIBLE);
+    complete(&create->started);
+    schedule();
+
+    if (!kthread_should_stop())
+        ret = threadfn(data);
+
+    /* It might have exited on its own, w/o kthread_stop.  Check. */
+    if (kthread_should_stop()) {
+        kthread_stop_info.err = ret;
+        complete(&kthread_stop_info.done);
+    }
+    return 0;
```
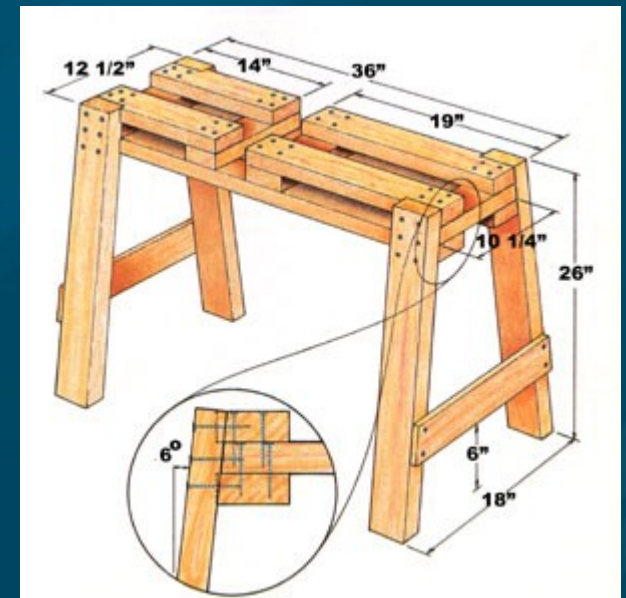
$\neq$

"the code is so beautiful!"

"does it do the job?"

Toolchains

- Complex problems need sophisticated tools

- Complex tools are built up from simpler tools

- Always know what's in your toolbox:
  the tools you have to tackle problems

  - In software: libraries

  - In math: axioms

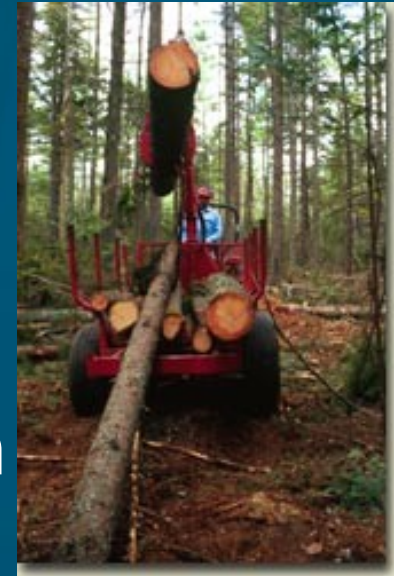 - In philosophy: worldview, context

- In 14x: Python + libraries

TRINITY
WESTERN
UNIVERSITY

Woodcutting example

- (see M2 text ch1.4, pp.4-5)
- **Problem**: cut 10 cubic metres of firewood
- Solution: 1st pass, 2nd pass, …
- What are the library functions used in each version?

TRINITY WESTERN UNIVERSITY

Woodcutting example

- We write out the solution in different levels of detail depending on
  - Who/what is executing the solution
  - What tools are available
- The solution is different for
  - An experienced lumberjack, w/good tools
  - A rookie who's never used a chainsaw
  - A software-controlled robot
  - A busy construction foreman
- *(which are more abstract / more concrete?)*

TRINITY
WESTERN
UNIVERSITY

Designing, not "hacking" code

- Look before you leap;
  think before you speak;
  design before you code!

- Programmer's optimistic schedule:

  - 4/5th coding

  - 1/5th testing/debugging

- Real-life schedule:

  - 1/3rd planning (***W***rite, ***A***pprehend, ***D***esign)

  - 1/6th coding (***E***xecute)

  - 1/2 testing/debugging (***S***crutinize)

Write
Apprehend
Design
Execute
Scrutinize

TRINITY WESTERN UNIVERSITY

Review

- **Toolsmiths** must know their **toolboxes**
  - (what does it mean for a computing scientist to be a toolsmith?)
- ***WADES*** *(Write, Apprehend, Design, Execute, Scrutinize)*
- **Top-down** vs. bottom-up
- First step in problem-solving? (don't code yet!)
- Levels of **abstraction** / levels of detail
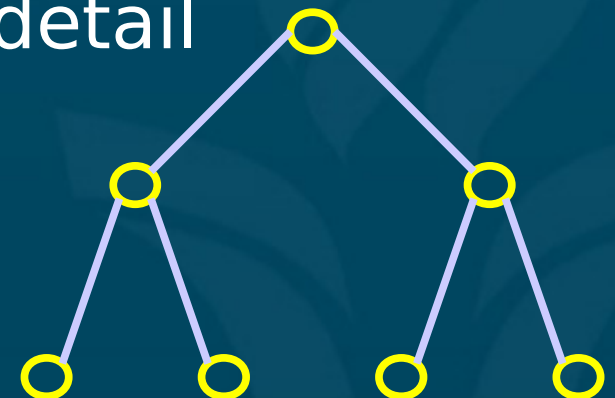
TRINITY
WESTERN
UNIVERSITY

Outline for today

- Course information

  - Course website, syllabus, schedule

- Guest alumnus: Kevin Tan / Accenture

- Programming as problem-solving

  - Tools, toolsmiths, toolboxes

  - Top-down vs. bottom-up design

  - Example: woodcutting

- Demo of our Python programming environment

- Tour of the CSI computing lab

TRINITY
WESTERN
UNIVERSITY

Python/IDLE demo

- (demo of the Python programming environment)

TRINITY
WESTERN
UNIVERSITY

Why Python?

- Why not M2, Java, C++, C#, PHP, Ruby, etc.?

- Syntax vs. semantics (more in a later section)

- At the CMPT14x level, the semantics of procedural programming in all these languages are pretty much the same

  - The only difference is syntax:
    ```
    for (i=0; i<10; i++) {          (C++)
    for i in range(10)              (Python)
    ```

- After this class, you'll be able to pick up any procedural language pretty quickly

TRINITY WESTERN UNIVERSITY

TODO items

- Familiarize yourself with the course website: http://cmpt140.seanho.com

- Do the Python/IDLE intro by next Thu (nothing to turn in, not graded)

  - Lab1 is due the following Thu after that

- Read ch1 of the textbook

- HW1 this Tues before start of class

  - Electronic turn-in: upload to myCourses

Tour of the CSI computer lab

- Neu20, also called "the senior lab"
- Only for students enrolled in a CMPT course
- Keepers of the Lab: Joel Schwartz, Dave Friesen
- Different login, network from rest of TWU
- Special key issued to you, tracks usage
- Food/drink are allowed; fridge/microwave/sink
- Security cameras go straight to recording; only looked at if something bad happens
- Don't make bad things happen; be good!

TRINITY WESTERN UNIVERSITY