CSE 303 Concepts and Tools for Software Development

Richard C. Davis
UW CSE – 12/8/2006

Lecture 25 – Closing Words

Administravia

- Papers due now
  - Late? E-mail to Richard Davis
- Review session
  - Sunday Noon-2PM, EE-037
  - You'll need your ID card to get in
  - Can't make it? E-mail Richard Davis.
  - I'll find another time for those people
    - Probably on Monday

Administravia (cont'd)

- Exam on Tuesday, 2:30 PM, this room
  - Bring a **TWO**-sided sheet of notes (8½" x 11")
  - How to study
    - Go over topic sheet (posted on web under "exams")
    - Go over lecture slides
    - Go over code studied in class
    - Go over assignments
    - Look over reading material
      - Only if you don't understand something from above
    - Do finals from previous two quarters
      - topics slightly different.

Where We Are

- Last time
  - Profilers
- Today
  - A few words about Performance Tuning
  - Closing Words
  - Course Evaluation

Poor man's Profiling with "time"

- Use to measure whole program time
  - Usage example: `time program args`
- Three types of time reported
  - real: roughly "wall clock"
    - Includes time used by other programs
    - Includes disk access time
  - user: time spent running code in the program
  - system: time the O/S spent doing stuff
    - Includes I/O (Note: `gprof` doesn't measure this)

Performance Tuning

- "First make it work, then make it fast"
- Steps in performance tuning
  1. Get things to work well, write clean code
  2. Locate bottlenecks and performance issues
     - Sometimes they are obvious
     - Otherwise, profiler can help
  3. Optimize your overall program
  4. Use low-level tricks if you need to
- Try compiling with "optimization flags"
  - `gcc`/`g++` uses `-O1`, `-O2`, `-O3`

What You've Learned

- Practical applications of computer science
  - Example: **gdb**
    - You know debugging concepts
    - You also know a handful of commands
  - CS knowledge != knowing tons of commands
    - But knowing some is very helpful
    - Always seek to learn more

Course Concepts

- Shell scripting
  - Automating program invocation
- C/C++ programming
  - A lower level of computing
  - Data is just bits
  - Pointers are explicit
  - Memory managed manually

Course Concepts (cont'd)

- The build process
  - How to manage source files
  - How they turn into running programs
- Software Engineering Methods/Tools
  - How to communicate with other developers
  - How to build robust programs
- Other Tools
  - How to measure running programs

Societal Implications

- We talked about it like computer scientists
  - We're logical and (usually) rational
  - We understand limitations of
    - Technology
    - Humans
- "We" should'nt make all decisions ourselves
  - But we should be actively engaged in process

You've made two steps toward this

Sony Pictures Imageworks -- Graphics Software Developer

Sony Pictures Imageworks, an Academy Award winning digital production studio, is seeking a software developer to design, develop and maintain lighting software for feature films. Our lighting software, Katana, is currently being used for our upcoming movies "Spiderman 3" and "Surf's Up". Candidate should have programming experience in one or more of the following areas: lighting, compositing, rendering.

Required skills:
* C/C++
* Linux (RedHat/Centos)
* Familiarity with Renderman or other production renderers.

Desirable skills:
* Python programming
* GUI programming (Qt/PyQt, WxWidgets)
* Knowledge of Maya API, Houdini
* Shader writing, compositing
* OpenGL and 3D math.

The Ultimate Goal

- In 5 years, this class will look like a waste
  - You won't remember not knowing this
  - You won't know why you need a class
    - To pick up a new tool
    - To automate a repetitive task

What should you learn next?

- Plenty more about each topic
- Plenty of other topics
  - Concurrency
- Some interesting things
  - Scripting languages: Perl, Python, Ruby
  - Managing dependencies (`gcc -M -MM -MG`)

What should you learn next? (C++)

- Mozilla Portability Guide
  - Avoid templates
  - Don't use exceptions
  - Don't use STL (or ostream) (try www.boost.org)
  - Don't use namespaces
  - Careful with `for (int i=0;` …
  - Always declare copy constructor & operator=
- http://www.mozilla.org/hacking/portable-cpp.html

How should you go about learning?

- When curious about topic "X"
  - Search for "X", "X tutorial", "X documentation"
  - Plenty of decent tutorials out there
- Learn from each other
  - Better to ask a question today than tomorrow
  - Books are good, too
- The more you know, the easier it gets

How should you go about learning?

- When curious about topic "X"
  - Search for "X", "X tutorial", "X documentation"
  - Plenty of decent tutorials out there
- Learn from each other
  - Better to ask a question today than tomorrow
  - Books are good, too
- The more you know, the easier it gets

The End

- That's All
- Now for a course evaluation

13–15) What I have learned in _____ will be useful for my future work in computer science.

13) lectures
14) programming assignments
15) social implications assignments/discussions

16) Programming assignments were interesting.

Strongly Disagree 0
Disagree 1
Somewhat Disagree 2
Somewhat Agree 3
Agree 4
Strongly Agree 5

17) HW1–HW4 (Unix, C) were…

18) HW5–HW7 (C++, Tools) were…

Much Too Hard 0
Too Hard 1
Just Right 2
Too Easy 3
Much Too Easy 4

On Back of yellow sheet:

Which societal implications discussion format did you find the most informative? Why?

- Group discussion + present summary
- Group discussion + argue opposite position
- Full class discussion