Follow the River and You Will Find the C

A systems programming course with a narrative

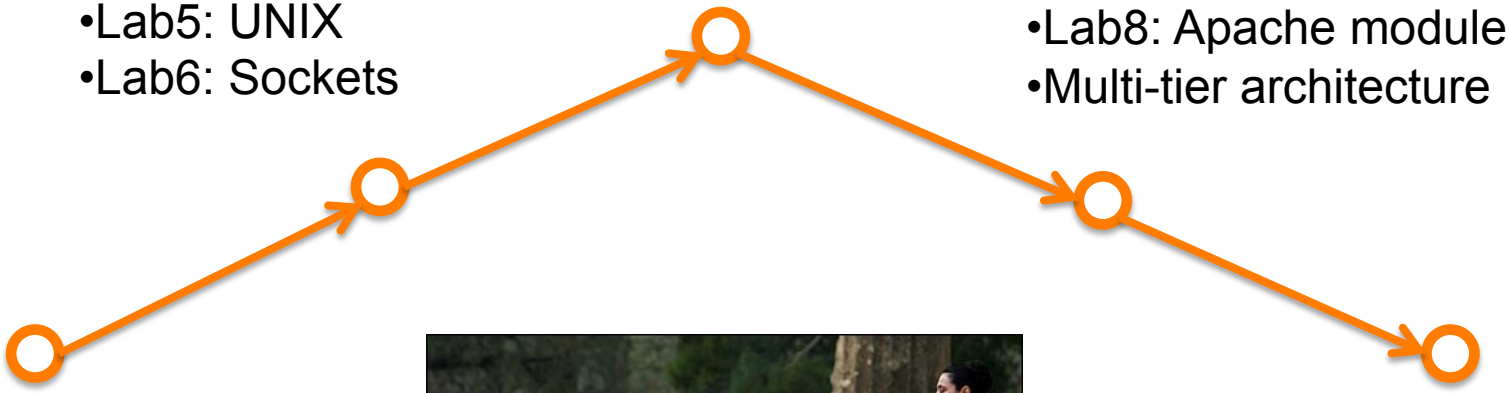
Overview of W3157 Advanced Programming

The course, a drama

Climax
•Lab7: Web server

Rising action
•Lab4: I/O
•Lab5: UNIX
•Lab6: Sockets

Falling action
•Lab8: Apache module
•Multi-tier architecture



Exposition
•Lab1: Tools
•Lab2: Pointers
•Lab3: Linked List

Resolution
C++ essentials
•Lab9: Object lifetime
•Lab10: Linked List II
•SmartPtr

Lab 1 - 4: C language

- Objective
  - Learn C, top to bottom

Lab 5: UNIX Processes (1)

- Objective
  - Understand processes in UNIX
  - Understand (conceptually) TCP/IP networking
- What we learned in the lectures
  - Role of Operating System
  - UNIX I/O using file descriptors
  - Process control using fork() & exec()
  - TCP/IP networking overview
  - Inter-process communication primitives:
    - Signal, Pipes, Sockets

Lab 5: UNIX Processes (2)

- What we did in the lab
  - 1a) Created mdb-lookup network server without socket programming
    - Composed a chain of pipeline using named pipe and netcat
  - 1bc) Wrote a control program that fork(), exec(), and waitpid() the mdb-lookup network server
  - 2) Created orphan and zombie processes
- Final exam study focus
  - Have a clear understanding of UNIX process hierarchy (part 1bc and part 2)
  - Understand the difference between the different inter-process communication primitives
  - Understand how the pipeline from 1a) worked
- For further study after this class
  - W4118: Operating Systems
  - *Advanced Programming in the UNIX(R) Environment* by Stevens
    - The authority on this topic
  - *Understanding UNIX/LINUX Programming: A Guide to Theory and Practice* by Bruce Molay
    - A lighter and friendlier treatment of the topic

Lab 6: Sockets API and HTTP protocol

- Objective
  - Learn sockets API
  - Learn HTTP protocol
- What we learned in the lecture
  - How to write a TCP client and a TCP server using sockets API
  - How HTTP works
- What we did in the lab
  - Wrote mdb-lookup network server for real, using sockets API
  - Wrote a HTTP client that downloads a URL
- Final exam study focus
  - Know the sequence of sockets API calls and what they do
  - Know the HTTP protocol
- For further study after this class
  - W4119: Computer Networks
  - *Unix Network Programming, Volume 1: The Sockets Networking API* by Stevens
    - The authority on this topic
  - *TCP/IP Sockets in C: Practical Guide for Programmers* by Donahoo and Calvert
    - A lighter and friendlier treatment of the topic
    - We used their sample code in class

Lab 7: Writing a web server

- Objective
  - Learn how web servers work and write one yourself!
  - Understand modern client-server architecture based on web
- What we learned in the lecture
  - How a dynamically generated web pages work
  - Overview of different software architecture
- What we did in the lab
  - Wrote a web server serving static pages
  - Wrote a web server serving dynamic pages (mdb-lookup)
- Final exam study focus
  - Know HTTP protocol and how to use it for dynamic contents
  - Revisit all versions of mdb-lookup from the software architecture point of view
- For further study after this class
  - W4111: Database Systems
  - W4156: Advanced Software Engineering

Lab 8: Writing an Apache module

- Objective
  - Hands-on experience building and installing open-source software
  - Understand what it means to write an extension for existing software
- What we did in the lab
  - Built and installed Apache web server
  - Wrote an Apache module for mdb-lookup
- Won't be on the final exam

Software Architecture: The Big Picture

- Retrace the evolution of MdbLookup
  - Lab4: command line, access local database
  - Lab5: server, put together with Netcat and pipes
  - Lab6: server, coded using the sockets API
  - Lab7: web-based server, written from scratch
  - Lab8: web-based server, written as Apache module
- Now you understand multi-tier client-server architecture
  - Underlying architecture for LAMP, J2EE, etc.

Lab 9: C++ basics

- Objective
  - Learn fundamental concepts in C++
- What we learned in the lectures
  - Stack v. Heap allocations
  - Pointer v. Reference
  - Basic 4
- What we did in the lab
  - Traced constructor and destructor calls in order to understand exactly when these are called
  - Implemented missing member functions in MyString
- Final exam study focus
  - Stack v. Heap allocations
  - Reference type
  - Basic 4 - IMPORTANT!
  - MyString - IMPORTANT!

Lab 10: Real-world C++

- Objective
  - Learn how to use templates and standard containers
  - Learn how to provide interface abstraction
- What we learned in the lectures
  - Templates
  - Standard sequential containers: string, vector, list, deque
  - Standard associative container: map
  - Advanced C++ usage: smart pointer using reference counting
- What we did in the lab
  - Wrapped a legacy linked list with a C++ class interface (part 1)
  - Wrapped a standard container to provide the same interface as part 1
- Final exam study focus
  - Understand what we had to do for part 1 and why
  - Understand templates (not the detailed syntax though)
  - Understand standard containers
  - Understand iterators
  - Understand SmartPtr
    - How reference counting works
    - How SmartPtr is used

Come full circle – Java-style object reference in C++

- "I miss Java…"
  1. Nice Java code

     ```
     Foo b = a.createFoo();  b.doSomething(); return;
     ```
  2. Same exact code in C++ (or is it?)

     ```
     Foo b = a.createFoo();  b.doSomething(); return;
     ```
  3. We can do this, but…

     ```
     Foo *b = a.createFoo();  b->doSomething(); return;
     ```
  4. Now this come pretty darn close

     ```
     SmartPtr<Foo> b = a.createFoo(); b->doSomething(); return;
     ```

- SmartPtr
  - Reference-counted, so can be freely copied
  - Initialized with pointer to heap-allocated object
  - Overloads operator->() and operator*()

Congratulations!

- Enjoy your accomplishments – big and small
  - Do not compare with peers
  - Compare with yourself 12 weeks ago
- Stress? What stress?
  - Have fun; be happy; be honest to yourself
  - Life is long. Really.

Beyond 3157

- Fill out CourseWorks evaluation
- Please do NOT share your materials
- Keep programming!

All good things…