Goals

Understand programming better
- Linking
- Exceptions
- Memory
- I/O
- Networking

Prepare for systems classes
- Computer architecture
- Compilers
- Operating systems
- Networking
Computer Organization

Hardware/software interface (ELEC 220)
- Basic hardware organization of computer systems
- Assembly language
- How low-level software manipulates hardware state

System/application interface (COMP 321)
- Abstract hardware organization
- C language (one step above assembly language)
- How to use operating system services to access system resources
Who are we?
Who are you?
Why use C?

Interfacing with run-time and operating systems is more suited to low-level programming.

Much of the software that you use is written in C.

Understanding C and assembly is key to understanding how programs execute:

- Behavior of programs in presence of bugs
  - High-level language model breaks down
- Tuning program performance
  - Understanding sources of program inefficiency
- Implementing system software
  - Compiler has machine code as target
  - Operating systems must manage process state
Problems in Low-Level Programming

It’s easy to make mistakes
When you do, you’re out of luck

Imperative programming

Few abstractions
• No objects, abstract functions, ..

No safety net
• Direct access to system resources
• Manual memory management
Programming Survival Skills

Planning

Good style & documentation

Defensive programming

Debugging
What this course is *not*:

This is not a course about the art of programming
- Other courses have/will cover programming principles (e.g., COMP 215, 310, …)

This is not a course about the C language
- You will gain a familiarity with C
- There are a lot of C concepts that we will not cover
- We will not focus on large-scale design in C

C is simply a useful vehicle for learning system-level concepts
Course Perspective

Upper-level systems courses teach how systems work so you can *build* them

- **Computer architecture**
  - How does a microprocessor work?
- **Compilers**
  - How does a compiler work?
- **Operating systems**
  - How does an operating system work?
- **Networking**
  - How do network protocols work?
Course Perspective

This course teaches how to use systems

• Purpose is to show how by knowing more about the underlying system, one can be more effective as a programmer

• Enable you to
  • Write programs that are more reliable and efficient
  • Incorporate features that require hooks into OS
    – E.g., concurrency, signal handlers

This course covers some material that you won’t see elsewhere
Syllabus Overview

Machine-level representation of programs
  • Assembly to C

Linking and Virtual Memory
  • How does a program actually get loaded and run?

Exceptions
  • Critical events can happen outside your program

I/O, Networking, and Concurrency
  • Programs must communicate to be useful
Linking

Libraries

• What are they?
• How are they used by your program?

Incremental compilation

• How can you minimize recompilation?
• How are 1000s of files combined into one program?

Errors

• Why does your program compile but not run?
• Why does Windows tell you that “<blah>.dll is out of date”?
Virtual Memory

Address spaces

- What does a program’s address space look like?
- How are programs loaded into memory?
- How are programs isolated and protected from each other?
- How do programs share memory?

Allocating memory

- Where is your program’s data stored?
- What if you need more memory?
- How is this allocation managed?
Exceptions

Running programs
- How does the shell work?
- What is a process, and how is it created?

Communicating with a program
- What happens when you type Ctrl-C, Ctrl-Z, etc.?
- How does your program find out about external events?

System calls
- What happens to your program when you invoke the operating system?
I/O, networking, and concurrency

I/O interface
- How are files read, written, shared, etc.?
- Which I/O functions should you use and why?

Network access
- Who manages the network?
- How can your program access the network?

Concurrency
- What are some of the problems with concurrency?
- How are concurrent programs written?
Exposure to Real Programs

Over the course of the semester you will be exposed to several “real” programs

Shell

• You will write pieces of a functioning Unix shell

Dynamic memory allocator

• You will write a functioning memory allocator

And more...
Logistics

Lectures: T and Th 2:30-3:45 Keck Hall 100
Labs: M 4:00-5:30, T 4:00-5:30, or W 3:00-4:30 Symonds II Lab
Lecturers: Alan L. Cox and Michael Fagan
TAs: Melinda Crane, Riley Gibson, Yizi Gu, Dhruv Madhok, Kyler Mejia, Visit Pataranutaporn, Luke Samora, Susan Wen, Yufeng Zhou, and Jiafan Zhu
Webpage: http://www.clear.rice.edu/comp321/
Announcements: On Piazza
Textbook: Computer Systems: A Programmer’s Perspective, 3rd Ed. by Bryant and O’Hallaron
Weekly Labs

Emphasis

- C programming, debugging
- In-depth hands-on exercises
- General programming tips
- Other cool topics in computer systems as time permits

Requires access to CLEAR servers

- Go to http://apply.rice.edu/ for an account
Labs

Everyone should attend

- You should treat labs like lectures
- Many key concepts needed by assignments will be covered in labs
- Especially crucial for those who have never programmed in C
- And for those who have not programmed in a Unix environment

Opportunity to ask questions

- Instructor and TAs will be available
- You might learn some unexpected neat tricks!
Assignments/Projects

Assignments all involve programming
- 6 assignments throughout the semester
- First 2 are to get you familiar with C programming
- Last 4 are to teach the course concepts and assume you are comfortable with C
- An introductory book on C may be helpful, e.g., *Head First C*

Comprehensive take-home exam during finals period
- No quizzes or exams during the semester
Assignment Policies

Carefully read the assignments web page
  • Honor code policy
  • Slip day policy

All assignments will be posted on the web page

Assignments are due at 11:55PM on the due date, unless otherwise specified

Assignments must be done on CLEAR servers
  • Other systems may behave differently!
Next Time

Begin introduction to C

* Lab this week will show some basic C programs
* Start with simple data types
* First programming assignment to get everyone familiar with C