CSci 4061

Introduction to Operating Systems

Administrivia, Intro
Welcome to 4061!

• Me:
  – Jon Weissman
  – CS Professor circa 1999
  – Call me “Jon”

• TAs:
  – Zach Leidall, Francis Liu
You

- CS BS majors
- CS BA majors
- CS minors
- CompE majors
- EE majors
- EE MS
- EE PhD
- ITI BAS majors
- Honors undergraduates

- Other and have a valid reason: 
  advisor@cs.umn.edu
Getting In
Logistics

• Lecture
  – Th 6:30 - 9, KH 3-230

• Jon’s (jon@cs.umn.edu) office hrs
  – TBD, KH 4-225D
  – Also come by when door is open
  – Can email for appointment other times

• TA office hrs: check course website
  – TBD, KH 2-209
Introduction

• CSCi 4061 is a rigorous course
  – Systems programming focus

• Expected background
  – CSCi 2021 (Machine Org and Arch)
  – CSCi 3081 (C/C++, even better)
  – Know how to edit, program, debug on preferably Linux systems
  – Can program in C or learn really fast
    • This is where we will lose people ... rapidly
Survey Time
Course Outcomes

You will learn how to:
• Write code that exploits OS features
• Write code that is efficient, reliable, and possibly secure

You will also learn a little bit about OS internals but mostly the EXTERNAL interface

You will learn about the UNIX/Linux interface but not every boring parameter setting

You will learn about general systems programming concepts beyond just OS
Why C?

• High-level languages are too far away from the machine

• Examples of applications that must be fast and use low-level OS facilities:
  – JVM
  – Web browser/server
  – DB engine
  – Text editors
  – on and on
Android (anecdotal)

- Sensor application
  - gcc (C) compiled code takes X time
  - Java compilation: 8-18X time
Our Perspective on OS

Two views

– conceptual view: what is inside the OS?
– user view: what can the OS do for me?

User view focus

– Abstraction
– APIs
– Libraries
To Be Successful in 4061 ...

• Be able to hunt down materials on your own (beyond the book)

• Be willing to learn by doing

• Be able to work effectively with others

• Ask questions
To Fail ....

• Rarely come to class

• When you do: be disruptive, sleep, surf

• Always seek to find solutions elsewhere before trying things on your own

• Succumb to cheating
  – We will be running checking software
Class Structure

• Main lecture
  – Motivate you
  – Cover concepts and abstractions
  – Provide examples and use cases

• Recitation
  – Hands-on C and UNIX/Linux
  – Some review (initially some C and UNIX)
  – Every section is identical
Class Resources

- **Web page** obviously
  - Information *(read it this week)*
  - Lectures (or sketches thereof)
  - Projects
  - Forum
  - Dates *(exams are tentative!!!)*

- Other useful Web links
- Textbooks
Books

• **Required:** Unix Systems Programming: Communication, Concurrency, and Threads, Robbins and Robbins (*R&R*)

• **Optional (inside-view):**
  – Operating Systems Concepts, Silberschatz, Galvin, and Gagne (*S&G*)
  – Modern Operating Systems, Tanenbaum (*MOS*)

• **Optional (Systems Programming):**
  – Unix Systems Programming, Haviland et al
  – Advanced Programming in the UNIX Environment, Stevens

• **Optional (C programming):** see class website
Brass Tacks: Coursework

• Four-Five systems programming assignments
  – Teams of 3 (composition TBD)
  – About 2 weeks per lab
  – Electronically submit and we’ll run it (CSE lab machine)
  – May provide test cases
  – Everyone gets same grade
  – If partner is slacking tell us immediately
Groups

• If you are very experienced ...
  – Be willing to take on someone less experienced

• If you are not very experienced ...
  – Be willing to approach someone more experienced

• Do not want to see bi-modal groups
  – Will ask TAs to break them up
Coursework (cont’d)

• Exams to test conceptual material and programming skill
  – correlate lab performance with exam
  – make ups? (don’t go there) unless your cat is on fire

• Late project work
  – NONE

• Re-grading?
  – 1 week window from return date
ADMIN Questions?
Topics

- OS Overview
- Programs and Processes
- I/O and devices
- File systems
- Communication
- Exceptions
- Threads
- Synchronization
- Memory Management
- Network programming
- System Design
Cross-cutting theme 1

• Concurrency
  – activities (resource sharing) appearing to occur at the “same time”:
    processes, threads, synchronization
Cross-Cutting Theme 2

• Asynchrony
  – dealing with unpredictable events (in time): exceptions, devices, I/O
Cross-Cutting Theme 3

- Communication
  - information transfer: communication, network programming
What is an OS?
Operating Systems: Two Interfaces

• The operating system (OS) is the interface between user applications and the hardware.

• An OS implements a virtual machine that is easier to program than the raw hardware
  – Example?
Operating System Roles

• Referee
  – Resource allocation among users, applications
  – Isolation of different users, applications from each other
  – Communication between users, applications

• Illusionist
  – Each application appears to have the entire machine to itself
  – Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport

• Glue
  – Libraries, user interface widgets, drivers, ...
Example: File Systems

• Referee
  – Prevent users from accessing each other’s files without permission
  – Even after a file is deleted and its space re-used

• Illusionist
  – Files can grow (nearly) arbitrarily large
  – Files persist even when the machine crashes in the middle of a save

• Glue
  – Named directories, printf, ...
Next Time

Read Chapter 1 (R&R), opt: Chapter 1 (MOS) or Chapters 1 and 2 (S&G)

Explore website: schedule, dates, syllabus

Next topic: OS Concepts and Structures