CSci 4061

Introduction to Operating Systems

Administrivia, Intro

Welcome to 4061!

- Me:
  - Jon Weissman
  - CS Professor circa 1999
  - Call me “Jon”
- TAs:
  - Kwangsung Oh
  - Jaskaran Veer Singh
  - Shalini Pandey
  - Aravind Alagiri Ramkumar
  - Meng Zou
  - Shaurakar Das

Getting In

- If students drop, there could be space, check by next Tuesday
- If you are planning to drop, please do it ASAP

Logistics

- Lecture(s)
  - Tu/Th 1-2:15 (Bru 230); 4-5:15 (Tate 105)
- Jon’s (jon@cs.umn.edu) office hrs
  - Office hours: see website
  - Also come by when door is open
  - Can email for appointment other times
- TA office hrs: check course website
  - TBD, KH 2-209

Non-Tech Interests

- Cycling
- Hiking
- Nordic skiing
- Film
- IPAs

Introduction

- CSCi 4061 is a rigorous course
  - Systems programming focus

- Expected background
  - CSCi 2021/EE 2361 (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

- How many have taken CSCi 2021 (machine organization) or equivalent?
- How many have taken CSCi 3081 (C/C++)?
- How many?
  - Experienced C programmer (> 200 lines of code, multiple modules, pointers, malloc)
  - Competent C programmer (50-200 lines of code, 1-2 modules, makefiles)
  - Novice C programmer (10-50 line program)
  - No experience at all

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
  - Any app that needs direct hardware access (screen, camera, audio, …)
  - 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

Who Does Well?

- Seniors generally fair the best

Class Structure

- Main lecture
  - Motivate you
  - Cover concepts and abstractions
  - Provide examples and use cases
  - Material that differs from posted slides looks like this

- Recitation
  - Hands-on C and UNIX/Linux
  - Some review (initially some C and UNIX)
  - Project checkpoints
  - 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)
  - <website>

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

- Optional (Systems Programming):
  - **Linux Systems Programming by Love**
  - Unix Systems Programming, Haviland et al
  - Advanced Programming in the UNIX Environment, Stevens

- Optional (C programming): see class website

Brass Tacks: Coursework

- Four 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 (In Progress)

- 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

- 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 (2 weeks)
- File systems
- Communication (2 weeks)
- Exceptions
- Threads (2 weeks)
- Synchronization (2 weeks)
- Memory Management
- Network programming
- System Design (maybe)

Cross-cutting theme 1

- Concurrency
  - activities (resource sharing) *appearing* to occur at the "**same time**":

    processes, threads, synchronization

  Examples from daily life?

  concurrent: "taking CS, Math, English courses in Spring 2016" (brain)

  parallel: "washing dishes and listening to ipod" (hands vs. ears)

Cross-Cutting Theme 2

- Asynchrony
  - dealing with **unpredictable** events (**in time**): exceptions, devices, I/O

  Examples from daily life?

“when mom wants to talk, my phone rings”

Cross-Cutting Theme 3

- Communication
  - information **transfer**:
    communication, network programming

“Jon -> Lab Assignment -> Submit Solution”

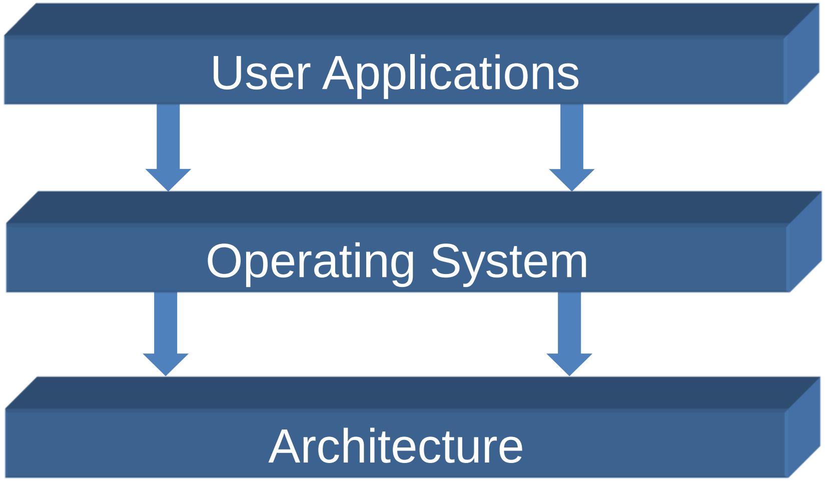
What is an OS?

Stakeholders?

- User
- System
- Which may be at odds?

Operating Systems: Two Interfaces

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



User Applications

Operating System

Architecture

- 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
- Illusionist
- Glue
  - Named directories, printf, …