1 Lecture: Introduction and Background

Outline:
Robust Programming
Background: The Ballad of the Unknown Stuntman
Syllabus
Preparation
How to succeed in this course
The Last Page
Foundations
  What an operating system is
  Virtualization and Transparency
  What an operating system is not
Major subjects for the quarter
This quarter

- Check on the book. (Third edition, Racoons)
- Make sure your lab accounts work.
- Assignments and labs are out
  - Coming attractions:
    - Use your own discretion with respect to timing/due dates.

- Office hours
  - Come
  - With high probability they are:
    - Monday: 3:10pm–4:00pm\(^1\)
    - Tuesday: 9:10am–10:00am\(^1\)
    - Wednesday: 3:10pm–4:00pm\(^1\)
    - Thursday: —
    - Friday: 3:10pm–4:00pm\(^1\)
  - These may be changed as things get sorted out.
  - Also lab times
  - If you’ve been through this before unsuccessfully, tell me. I can ask you awkward questions that’ll improve your odds of never doing 453 again.
  - A note on persistence

1.1 Robust Programming

Do it.

\(^1\)Office hours are guaranteed until the earlier of the posted end time or the time at which there are no more students.
1.2 Background: The Ballad of the Unknown Stuntman

Usually—if all is going well, that is—we aren’t even aware of the operating system. The operating system of a computer is the group of stagehands making everything work. Without them, the show couldn’t go on, but you only notice if something doesn’t work.

“The OS is our enemy?”

Have you ever wondered why your PC won’t speak to your printer even though the printer seems to be being extremely polite?

1.3 Syllabus

- The Course Staff
  me: Phillip Nico
  pnico@calpoly.edu
  Office: 14-205

- About this course:
  What is this course about?
  Why Study OS?
    – Because it explains the magic
  What does that mean?
    – OS Theory:
      * Concurrent programming (deadlock avoidance)
      * Scheduling algorithms (fairness, starvation)
      * Resource allocation (e.g., Memory allocation (paging) algorithms)
    – OS Implementation: How it’s really done (The lab)
    – Stark Raving Paranoia: The OS can’t stop, can it?
    – Course Objectives
      The purpose of this course is to gain experience with and understanding of operating systems principles and implementation. In the process, you will:
        * Examine the requirements of a modern operating system, including the fundamental problems of managing concurrent processes.
        * Understand the system call interface to an operating system.
        * Understand how an operating system gets started (boots) and takes control of the machine.
        * Understand the design and implementation of a filesystem.
        * Learn a bunch of other interesting things.

- Prerequisites:
  203 and (225 or 233) Know how to program and understand architecture.
  cpe 357 Be familiar with the C programming language, user-level unix, and the “user-side” of system calls.
Prerequisites. You must have satisfied the prerequisites. **If you have not, drop the class.** (Seriously.)

- Re-takes
  - If you are re-taking, perhaps talk to me.

- Enrollment:

  Current:

<table>
<thead>
<tr>
<th>sec 03</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrolled</td>
</tr>
<tr>
<td>target</td>
</tr>
<tr>
<td>waitlist</td>
</tr>
<tr>
<td>unique</td>
</tr>
</tbody>
</table>

  We’re gonna try something crazy and let the registration system do its thing. There are currently 19 open seats in other sections.


  Just about everything of interest ends up here, including:

  - The syllabus (that you should read)
  - Notes
  - Assignments (read these too. It helps.)
  - Labs
  - Solutions
  - (Probably) grades


  If you are unfamiliar with the UNIX system programming environment, you might also find the following helpful:


- Also: The Minix Programmer’s Manual (online—use the version on your machine)

  Note: “cmd(num)” means cmd as defined in section num of the manual.

- Grading policy:

<table>
<thead>
<tr>
<th></th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labs</td>
<td>15</td>
</tr>
<tr>
<td>Assignments</td>
<td>25</td>
</tr>
<tr>
<td>Midterm</td>
<td>25</td>
</tr>
<tr>
<td>Final</td>
<td>35</td>
</tr>
</tbody>
</table>
• Assignments: (Do Them)
There will be some. A variety of programming, problem set, and report writing, totaling 25% of the final grade.
I expect there will be 5–6 assignments over the quarter, making each one worth approximately 5%
I will publish weights, but reserve the right to change them.
Asgn1 is out already. GET YOUR ACCOUNTS NOW you can even use the lab time.
Deliverable: a program or a report.

Note on assignments:

– Do them!
– Try to break them
  * with real tests
  * on multiple platforms (e.g. 32- and 64-bit)
– If I provide a test harness, run it. It may not be my entire grading rig, but it’s probably a subset. (And it might be.)
– RTFA

• Labs (6–7):
Most weeks there will be lab exercises. These will be some combination of written problems and laboratory exercises.
Deliverable: usually a report, sometimes a program.

A note on the lab period.

• Policies regarding submission of work:

  – Late Policy
  – Submitting Work (paper, online (I'll check on this)) neatness.
  – Late days
  – “Good faith effort” required on all assignments. (Missing assignment reduces maximum possible grade.)

• Exams: a midterm and a final. Emphasis on problem solving.
The midterm is tentatively scheduled for may 8 (wed)

• Grading: Policy. 1-week deadline for regrades/errors.

• Class decorum

  – I encourage interaction in class. If you’re confused, so are other people.
  – Note
    * you all belong here.
    * Remember: of course you don’t know what you are doing. That’s why you’re in the class. Speak up.
– I will not penalize you for nonattendance, but you will lose sympathy in office hours.
   (Find out what happened)
– Please refrain from disruptive behavior.
– In particular, this quarter I am strongly encouraging no electronic devices in class. They’re just too distracting. (http://www.yorku.ca/ncedlab/laptopFAQ.html)

Laptops hinder classroom learning for both users and nearby peers

Faria Sana, Tina Weston, and Melody Wiseheart*

General Abstract: Laptops are commonly found in university classrooms. Thus, students will inevitably use laptop applications, such as games and social networking, during class time. To investigate whether multitasking on a laptop impedes in-class learning, we conducted two experiments in a simulated university lecture setting. We found that students who multitasked on a laptop during a lecture scored lower on a test compared to students who did not multitask, and students who were in direct view of a multitasking peer scored lower on a test compared to students who were not in view of a multitasking peer. The results suggest that multitasking on a laptop is a distraction to both users and fellow students and can be detrimental to learning of classroom materials. *formerly Nicholas J Cepeda

– I try to keep the two classes in sync, but I cannot guarantee it. I do ask students to attend their own sections for exams.

• Cheating: Don’t.
  – Integrity matters
  – Partnerships and proper collaboration.
    * DO: use each other for help
    * DO NOT: simply split up the work
  – Proper Collaboration
    * general principles/approaches
    * debugging consultation
  – Improper Collaboration
    * “Here’s my code”
    * specific solutions
  – Apply the “Competition” rule
  – Credit any work that comes from anyplace else. (Not doing your homework is better than getting caught cheating. Your name isn’t Tanenbaum, is it?)

The Pledge:
  – Don’t use others’ work while doing your own
  – Don’t make your work available. This includes taking reasonable precautions to prevent someone else’s taking it. E.g., not leaving your program lying around or leaving it online world readable.
It is never acceptable to allow someone else to have your source code for reference or to refer to someone else’s code while writing your own.

Cheating requires an “F” course grade

Not giving your work to others includes taking reasonable precautions to prevent others taking it.

1.4 How to succeed in this course

• Read assignments right away so they can be percolating.
• Do discuss problems with classmates and/or me
• Give yourself time to be stuck
• “I need insight now” is not effective
• When I give test harnesses, use them:
  – tryLab01 (don’t copy it)
  – tryAsgn1 (don’t copy it)
• It sounds silly to say, but: turn stuff in. (You can’t win if you don’t play).
• Read the textbook. Tanenbaum is a good writer.

1.5 The Last Page

Please fill this out as it helps me to know who you are, where you’re coming from, and what you expect from the course. It also helps me gauge enrollment.

1.6 Foundations

1.6.1 What an operating system is

Virtual Machine (Extended (abstract) machine) Simplified view of the system.

Operating Systems insulate users from the complexity of the underlying system and provide functionality that doesn’t exist.

(e.g. a filesystem: do you really care how a file is stored?)

(Do you want to know (or care) whether the floppy motor is on? The OS does. (and you do. You just want someone else to take care of it.))

Resource Manager Combination mediator and traffic cop.

(Consider printer spooling.)

Typically this is possible because only the OS has access to certain instructions. (kernel or supervisor mode)

This is just protecting from accidents, let alone malice.
The operating system is the one who *really knows what’s going on.* (and makes the magic happen)

That’s all well and good, but what *is* an operating system?

**a program.** That’s it.¹ Rather, it’s the first program that allows others to run. The things above are what makes an OS *useful.*

The OS can be considered the host of the party: it’s the first to arrive, the last to leave (after cleaning up), and mostly it tries to stay out of the way and keep the party going, but steps in where necessary to resolve conflict.

1.6.2 Virtualization and Transparency

From a user’s perspective this process reduces to *virtualization* and *transparency*

**transparent** It’s there, but you can’t see it.

**virtual** you can see it, but it isn’t there.

1.6.3 What an operating system is not

- Compilers
- Editors
- Command interpreters
- **Web Browsers(!)**

<table>
<thead>
<tr>
<th>Compilers</th>
<th>Editors</th>
<th>Shells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine Instructions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical Hardware</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.7 Major subjects for the quarter

Three areas I really want to develop this quarter:

1. OS Theory: How an we solve various problems well?
   - Concurrent programming (deadlock avoidance)
   - Page algorithms
   - Scheduling algorithms (fairness, starvation)

2. OS Implementation: How it’s really done (The lab)

3. Stark Raving Paranoia: The OS can’t stop, can it?

¹This answer would be considered correct, but not complete were I to ask the same question on a midterm. :-)

17
1.8 This quarter

What we’re looking at:

• How the OS does its magic
  General areas:
  – Processes and Concurrency
  – IO
  – Memory Management
  – Filesystems
  – Security?

• This class:
  – Why study OS?
  – Why Unix? — innards are exposed
  – Why Minix? — there aren’t too many innards
    * only 30,000 lines of code (vs. ≈9 million for Linux)

• Resources:
  – the CSL machines