

# 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
- About the lab
- About the assignment

- Check on the book. (Third edition, Racoons)
- Make sure your lab accounts work.
- Assignments and labs are out
  - Coming attractions:

Event	Subject	Due Date			Notes
asgn2	LWP	Mon	Jan 26	23:59	
asgn3	dine	Wed	Feb 4	23:59	
lab03	problem set	Mon	Feb 9	23:59	
midterm	stuff	Wed	Feb 11		
lab04	scavenger hunt II	Wed	Feb 18	23:59	
asgn4	/dev/secret	Wed	Feb 25	23:59	
lab05	problem set	Mon	Mar 9	23:59	
asgn5	minget and minls	Wed	Mar 11	23:59	
asgn6	Yes, really	Fri	Mar 13	23:59	
final (sec01)		Fri	Mar 20	10:10	
final (sec03)		Fri	Mar 20	13:10	

Use your own discretion with respect to timing/due dates.

- Office hours
  - Come
  - With high probabability they are:
    - Monday: 10:10am–11:00am<sup>1</sup>
    - Tuesday: 12:20pm–1:00pm<sup>1</sup>
    - Wednesday: 10:10am–11:00am<sup>1</sup>
    - Thursday: —
    - Friday: 10:10am–11:00am<sup>1</sup>

---

<sup>1</sup>Office hours are guaranteed until the earlier of the posted end time or the time at which there are no more students.

- 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.
- Forum (linked from web page)
- A note on persistence (tortillas?)
- Expect to fail. A lot.

## 1.1 Robust Programming

Do it.

## 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 (out of date):

	sec 01	sec 03
enrolled	30	30
target	30	30
waitlist	10	10
unique	20	

We’re gonna try something crazy and let the registration system do its thing, then talk about it.

- Web Site: <http://www.csc.calpoly.edu/~pnico/class/now/cpe453>

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

- Texts: Andrew S. Tanenbaum and Albert S. Woodhull. *Operating Systems: Design and Implementation*, third edition, Prentice-Hall, 2006. ISBN 0-13-638677-6

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

- W. Richard Stevens, *Advanced Programming in the UNIX Environment*, The Addison-Wesley Professional Computing Series, 1/e, Addison-Wesley, 1992.
  - Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, Third edition, Prentice-Hall, 2006.
- 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:
 

Labs	15%
Assignments	25%
Midterm	25%
Final	35%
  - 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.). These are located in `~pn-cs453/demos`, usually named `tryAsgnN` or `tryLabN`.  
Don't copy it. Run it from there. (Remember, to run something you name it.)
    - 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(not this quarter), 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 7  
The final? We’ll see.
- 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.
  - 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.
  - Spring quarter was a horrorshow
  - 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?)

Thought:

- If the only penalty is a bad grade...

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.<sup>1</sup> 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(!)**

Compilers	Editors	Shells
Operating System		
Machine Instructions		
Physical Hardware		

## 1.7 Major subjects for the quarter

Three areas I really want to develop this quarter:

1. OS Theory: How can 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?

---

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

## 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.  $\approx 9$  million for Linux)
- Resources:
  - the CSL machines

## 1.9 About the lab

Remember `man name(section)`

- `pipeit (ls | sort -r > outfile)`

What does this mean?

- There are three processes here
- You are the plumber/reaper
- This demonstrates the process abstraction that an OS provides. All communication and synchronization takes place through the OS.
- These must be concurrent (why?)

## 1.10 About the assignment

- `malloc(3)`: How does it work?
- libraries: Two forms
  - static (`libmalloc.a`)
  - shared object (`libmalloc.so`)
    - \* `LD_LIBRARY_PATH`
    - \* `LD_PRELOAD`

- don't call `sbrk(2)` for every call to `malloc(3)` (quilting analogy)
- remember how pointer arithmetic works (in the size of the pointee)
- `uintptr_t` from `<stdint.h>`
- About that Makefile...
- Note: the order of link commands matters to `gcc`

- Also Note: setting environment variables:

[ba]sh	<code>VAR=value</code>
	<code>export VAR</code>
[t]csh	<code>setenv VAR value</code>