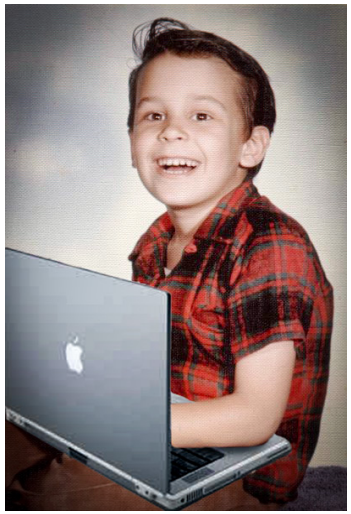
Introduction to Operating Systems CMPS 111, Spring 2015

Prof. Darrell Long
Kumar Malavalli Professor



Welcome!

Prof. Darrell Long
Office: E2-371
Hours: 1430-1530 MW
darrell@ucsc.edu

Subhag Ragi
Office: E2-380
Hours: Mon 2:30–3:30
Tue 11–noon
sragi@ucsc.edu

- Laboratory sections (all in Engineering 2 194)
  - Wednesday 11:00AM-12:10PM
  - Friday 08:00AM-09:10AM
- Class home page:
  - https://courses.soe.ucsc.edu/courses/cmps111/Winter16/01
  - https://classroom.google.com/c/NzU2MDcyMjcy
- Git page
  - https://git2.soe.ucsc.edu/users/git/

This is a hard class

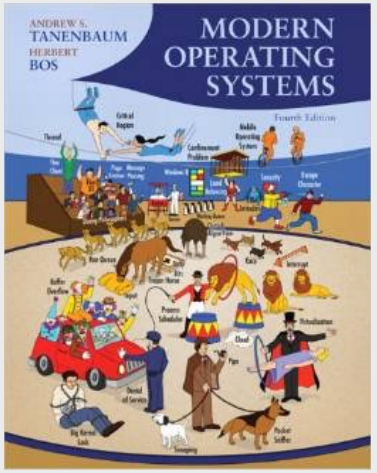
Course topics

- Introduction, concepts, review & history
- Processes
  - Synchronization
  - Scheduling
  - Deadlocks
- Memory management, address translation, and virtual memory
- Operating system management of I/O
- File systems
- Virtualization
- Multi-core systems
- Security & protection

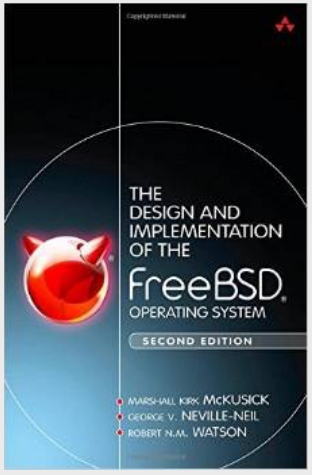
Books

Required

Modern Operating Systems, 4th edition (Tanenbaum)



Optional



The Design and Implementation of the FreeBSD Operating System, 2nd edition

Course requirements

- Two examinations
  - Midterm in the 5th–6th week
  - Final exam
- Projects
  - 1 pass/fail (check-off) project to get your environment set up
  - 4 graded projects during the quarter
  - About 2 weeks per project
- Notes
  - Notes submitted each Saturday
  - Graded pass/fail (check-off)
  - *May be brought to examinations*
- Class participation

Programming projects

- Modify FreeBSD
  - Runs on x86 hardware
  - Virtual machine software runs on OS X, Windows, Linux
  - Tool set runs on FreeBSD
- Learn about operating system structures
- Learn how to modify existing code

- Implement some of these:
  - Shell
  - Synchronization
  - Scheduling
  - System calls
  - Memory management
  - File system
- Learn how an OS really works!

Project logistics

- For each project, commit (using `git`)
  - Detailed design description
  - Code files & Makefiles used to implement the project
  - Files used for testing your implementation
  - Documentation on how to build, run and test the project
- *All* material must be turned in using `git`
  - Work may be done anywhere, including your own computer
  - Must be pushed to git2.soe.ucsc.edu (details online)
- Details on how to install FreeBSD and VMware online
  - VMware is free to students (one-year license)
  - Download copy of FreeBSD install disk from UCSC server
  - Check out personal copy of source code using `git`

Programming operating systems

- Operating systems are unlike most other programming environments you've used
  - Many resources aren't available inside the OS
  - Modifying existing code, not writing *de novo*
  - Difficult or impossible to use an IDE for debugging
    - If you make a mistake, the *computer* may freeze up
- Solution: careful planning!
  - Understand existing code *first*
  - Plan your solution on paper before coding it up
  - Use a programmer's editor to write code (not TextEdit or WordPad)
  - Use `git` to keep multiple versions of your work…

Version control using `git`

- Version control allows you to efficiently keep multiple versions of your code
  - Covers the entire directory tree, not just single files
  - Very efficient: keep as many versions as you like!
  - Go back to previous versions or see differences between versions to understand why something works (or doesn't)
- Each student has his/her own repository on git2.soe.ucsc.edu
  - Check out initial version of FreeBSD source code from git server
  - Make changes and commit them locally
    - Each commit is timestamped when it's made locally
  - Push commits back to the server at some point
    - No need to do this after *every* commit
- Grading is done on the last commit on the server before the project deadline

Taking notes

- Taking notes is a required but small part of your grade
  - You cannot pass the class by simply reading the slides
  - Notes must be turned in each Saturday by 1700h using `git`

- Notes are not a collaborative activity
  - You are expected to attend class and take notes during every lecture
  - You can get together with other students for study sessions and bring your notes

Grading

- Final grades based on:
  - Projects: 45% — all graded projects weighted equally
  - Notes: 10% — required
  - Midterm: 15%
  - Final: 30%
- Approximate grade ranges:
  - A: 89% – 100%
  - B: 79% – 89%
  - C: 69% – 79%
  - D: 60% – 69%
- To pass the class, you must
  - Complete all examinations and projects
  - Have **at least** a 50% average on examinations and 50% average on projects
  - These are necessary but not sufficient to pass the class
    - Example: 51% on examinations and 51% on projects ⇒ no pass

Getting help

- This can be a tough class—get help if you need it!
  - The course staff (professor, TA) are here to help you learn the material
  - It's up to you to ask for help
- **Don't wait too long!**
- Ask questions in class
- Go to section
- Visit office hours (professor, TA)
- Ask general questions on the forum
  - Course staff will be answering questions here
  - Students in the class can answer as well
- Ask specific questions by electronic mail to course staff
  - Expect short answers, not long explanations

Discussion forums

- We're using piazza.com to host class discussion forums: sign up at URL listed in the syllabus

- Ask questions on the forum about material you don't understand
  - Lecture & text
  - Homework
    - Since it's ungraded, collaboration is OK
  - General questions about projects
- Students and course staff can post answers
  - May provide faster feedback than asking questions via email
  - Benefit from others' questions & answers

Secrets to success in CMPS 111

- Projects
  - **Start projects early!**
  - Write up your design document before writing code!
    - Spend less time writing code
    - Make it easier to get help from the professor and TA
  - Use debugging tools
    - Details in lab section…
- Do the homework to test your own knowledge
  - If you don't understand something, ask
  - OK to work together on homework—it's ungraded
- The best time to get help is **as soon as possible**
  - Waiting until the last minute won't leave enough time for us to help you
  - You can always finish early and take the last day off….

Academic honesty

- You are expected to adhere to the highest ethical standards
  - All work you submit must be your own
  - You must give credit where it is due

- **Plagiarism of any form is unacceptable!**

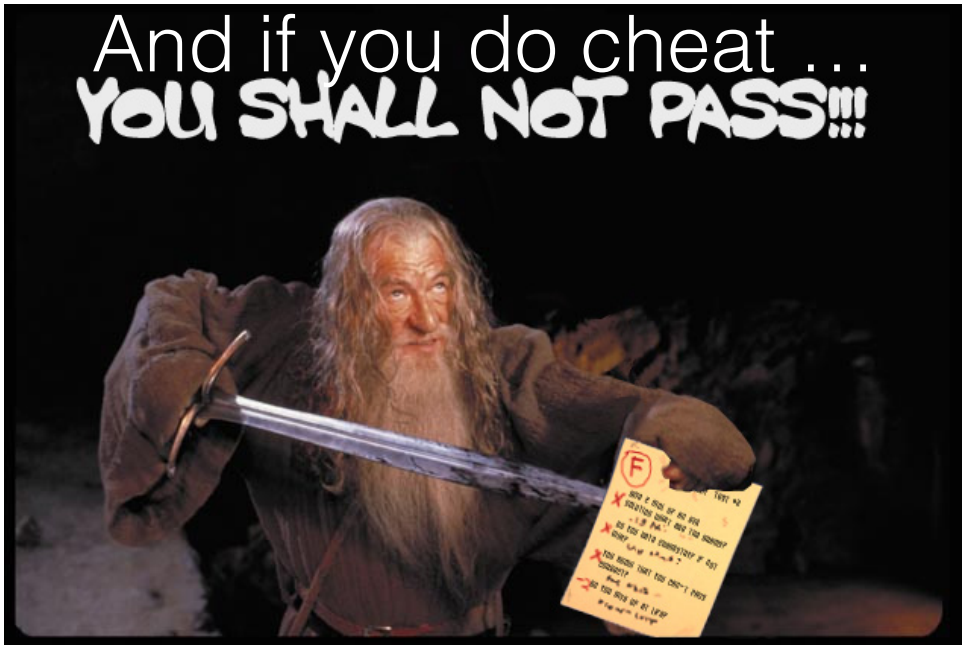
- Consequences of dishonest conduct
  - A letter will be sent to the School of Engineering and the provost of your college
  - You ***will*** fail the course

- Bottom line: **don't cheat!**

Cheating

You all *know* what cheating is …

but in case you do not, you *will* all be signing an acknowledgement regarding it.



What is cheating?

- Projects
  - You may collaborate as part of your assigned project group
    - You must complete the first graded assignment on your own
  - Collaboration with anyone outside your group is limited
    - Give credit to anyone from whom you get help
- Sample scenarios online at
  `https://classes.soe.ucsc.edu/cmps111/Spring15/academic_honesty.php`
- Examinations
  - You may not collaborate during an exam under any circumstances
  - Studying together before the examination is, of course, *encouraged*

- All students must turn in a signed sheet saying they understand the academic honesty rules
  - Must be done by the time you turn in Assignment 0

What to do after graduation...

- Graduate school or work?
  - Work: good if you want money now
    - Graduate school typically covers expenses and tuition, but you won't get rich there...
  - Graduate school: good if you like research (not being a code monkey)
  - Start now to apply for Fall 2017 (too late for Fall 2016)
    - Line up letter writers
    - Figure out where you want to go
    - Talk to (typically tenure-track) faculty!
- Either way, join the ACM / IEEE / USENIX
  - Community of colleagues
  - Access to papers
  - Informative (and fun) conferences
  - Cheap to join as a student!

Getting numbers right

- Many problems in computer systems involve numbers
  - How many disk requests per second?
  - How much memory?
  - How many interrupts can each CPU handle?
- Estimation can be useful to check your answer
- Example: how many disk requests can your five disk system handle per second?
  - Estimate
    - Disk requests take about 10 ms each
    - Each disk can do about 100 per second
    - Five disks can do 500 per second
  - Actual (tentative) answer: 54,000 requests per second
    - Is this likely to be right?

More on estimates

- Question: how much water flows out of the Mississippi River in a year?
- You could look the answer up online, but is it right?
- Solution: estimate
  - Two possible ways to get the answer
  - If they both agree (or are close), you're probably right…
  - The solution may not be in useful units (in this case, I found one in cubic feet per second)
- What are the two ways to figure this out?

- To avoid gross errors, you should know
  - Metric prefixes (kilo, milli, giga, etc.)
  - How to estimate using powers of ten (scientific notation)
  - How to convert powers of two to powers of ten