# Operating Systems


CSCI 4730 / CSCI 6730

Maria Hybinette



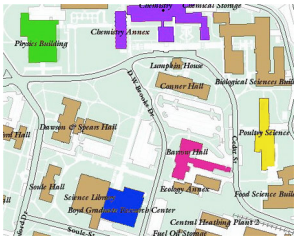
# Course Topic

- Today go over expectations and a a course plan
- Tuesday we will discuss presentation topics & some advice on giving talks



# Logistics

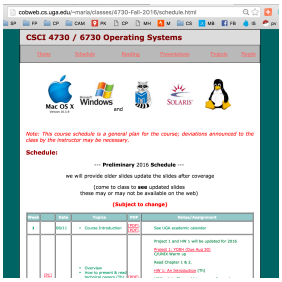
- Who am I?
  - Office: Boyd 219C
- Class:
  - Boyd 306 (blue)
  - Boyd 208 (blue)
  - Maria.hybinet
- Office Hours: Thu after class
  - And by e-mail appointment
- TA: TBD - check class web page for updates…






# Communication Links

- http://www.cs.uga.edu/~maria/classes/4730-Fall-2016/schedule.html
- Check often (everyday)
  - Your Responsibility
- Understand policies, honor code
- Work independently on projects and homework no line by line assistance –
  - No copying from anything
- Check page often for updates
- HW, Projects, Deadlines
- Email list: **Piazza**
  - Will set up (see web page for update) - We will use piazza
- Turnins:
  - Most likely ELC
  - May be via submit command (old style)






# Course Objectives

- Exposure to real kernel hacking.
  - And the concepts that make it work
    - Communication, Synchronization, Scheduling, Memory and Files
- Experience that carries to most other OS.
- Build appreciation for 'working' Operating Systems – commercial and 'free'
- Encourage
  - Continue help develop OSs over the net – join groups, hack kernel features and extensions.
- Improve your background when choosing a kernel to hack and work with.
- Introduction to research on operating systems: past and present.



# Areas that we will investigate

- Know and understand fundamental issues of operating systems
  - Processes (lecture, project, homework)
    Communication: Socket programming & other IPC
  - Threads (lecture, homework)
  - Scheduling (lecture, project, homework)
  - Synchronization & Deadlock
  - Memory Managements & Virtual Memory
  - File Systems
  - I/O System (presentations)
  - Mass Storage (presentations)
  - More…. Tune your programming skills and understanding -- resume building - simulation practice gives you -- versatility, Internet games, entertainment
    - Why learn programming when you can get a gorilla do it for you? [BONUS ?]

# Topics

- Know and understand fundamental issues of operating systems
- Processes (lecture, project, homework)
  - Communication: Socket programming & other IPC
- Threads (lecture, homework)
- Scheduling (lecture, project, homework)
- Synchronization & Deadlock
- Memory Managements & Virtual Memory
- File Systems
- I/O System (presentations)
- Mass Storage (presentations)
- More…. Tune your programming skills and understanding -- resume building - simulation practice gives you -- versatility, Internet games, entertainment



# Class Structure

- Read & Listen
- "Operating Systems Concepts." 9th Edition, Silberschatz, Galvin, Gagne (or later edition) *Required
- "Operating Systems, MINIX", Tanenbaum (Tentative)
- Practice
  - 5-7 programming assignments
  - At least 5, more likely 6.
  - Mini-Conference Technical paper presentations & summary.
  - Learn how to read/skim papers
  - present & listen to your peers
    - Technical papers and Tools Talk
  - Learn how to make a nice presentation - friendly environment
- Test
  - 2 Midterms, 1 Final, and Quizzes (**frequent**)



# Grading

- Theory 40%
  - 2 Exams (10% each) + Final 15% + Quizzes 05% = 40%
- Practice 5555555% (or 55%)
  - 9-11 homework (10%) & summaries (15%) & presentation (10%) & programming assignments (20%) & session chairing (HW)
- Participation 5%
- 100% attendance will raise your final grade by 2%
- We will use the College Board's convention to convert from percent grades to letter grades grades.
  - http://www.collegeboard.com/html/academicTracker-howtoconvert.html



# Policy on Collaboration

- Assignments/projects/summaries:
  - Purpose: familiarization of concepts and details of operating systems
  - Work on project independently:
  - No Direct Sharing of code
  - No line-by-line assistant
  - No exchange of code
  - **We will check this with software**
- You are encouraged to ask questions of one another, and to respond to other student's questions (and especially on the email list Piazza)
- Exams:
  - Closed-book. No outside assistance is permitted. No additional materials may be used.
  - **No make-up tests** unless absence is due to serious illness. Doctor's diagnostic note is required. The final grade will be scaled accordingly.



# Paper Presentations

- 1-2 presentations will be expected (2nd presentation may be a 'teams' of 2 presentations)
- We discuss topics, and tools.
  - Caveat: If someone sign up for a paper and then later drops, we may need to shift the last scheduled person to the empty slot(s) (other volunteers are welcomed and will be solicited in class).
  - Format:
- Presentations – 10 minutes long (about 10-15 slides)
- Core topics, research projects (e.g., clouds), project (MINIX) oriented topics, or Tools talks.



# Paper Summaries

- One page summary of an assigned technical paper -- need to reflect that you understand the paper and its contribution(s) to the area:
  - What is the problem that the authors are trying to solve? [why is it important]
  - What is their approach and how is it original and innovative? [compare against contemporary approaches]
  - How is the approach evaluated?
  - What are the assumptions/limitations?
  - Strength & weaknesses
  - What are the results/impact of paper (Why is this paper important, relevant)?
  - What constructive criticism can you give to the presenter (e.g., would should have been included/excluded)? Do not discuss presentation style of speaking, comment on 'content' of talk and possibly organization.

## Tentative projects for class

Tentative projects (these may change)

1. Simple shell/interpreter
2. A Gentle (tentatively) MINIX Kernel Hack
3. Process communication (Sockets)
4. Modify the MINIX Scheduler (Challenging)
5. Synchronization/Threads : Implement Semaphores in MINIX or Implement conditional variables.
6. Virtual Memory allocation policies in MINIX
7. File Server for MINIX (if time permits)



## Homework 1

- Will be listed Tuesday, and Due Thursday.



## Background

We will program in C, it is expected that you are a good programmer.

1. Have you taken a computer system course that surveys basic computer hardware and systems software components?
2. Have you taken a course that covers computer architecture topics? Do you understand the basics of how computer systems work?
3. Are you familiar with C? Have you programmed in a UNIX environment? This introductory lesson (next week) should give you a better idea of the practical skills you will need for this course.
4. Are you **interested** in understanding the internals of how computing systems work, and how to get them to work "better" (as opposed to being interested in upper-level software and building cool applications)?



## Course Contributors

- Tidbits & Material are drawn from several resources:
- Book Authors:
  - Avi Silberschatz, Peter Baer Galvin and Greg Gagne
  - Andrew S. Tanenbaum, Vrije Universiteit
  - William Stallings
  - Deitel & Deitel's OS Book
  - And others.
- Other Instructors & Colleagues:
  - Andrea & Remzi Arpaci-Dusseau, University of Wisconsin
  - Andy Wang, (UCLA) now Florida State University
  - Fred Kuhns, Washington University
  - Jeff Donahoo, Baylor University (TCP/IP and sockets)
- Students Feedback
- Wikipedia



## Quiz: C Practice

Please turn in on note book paper: Please tell us:

1. Name, major, year?
2. a) What computer hardware do you own (include smart phones if you own one)?
3. List familiar Operating Systems (indicate the OS on your laptop)
4. List the Operating Systems that are familiar to you?
5. Write a C program that computes the mean, mode (most frequent) and median (if even then take the average of the 'middle' two) of integers entered from standard input, one number per line, and let the number 0 indicate end of input). Assume the range of integers are between [0,1000] inclusive.

```
1. #include <stdio.h>
2. int inputnumber = 0;
3. scanf("%d",&inputnumber);
```