Operating Systems

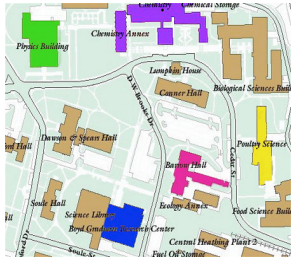
CSCI 4730 / CSCI 6730

Maria Hybinette

Course Topic

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

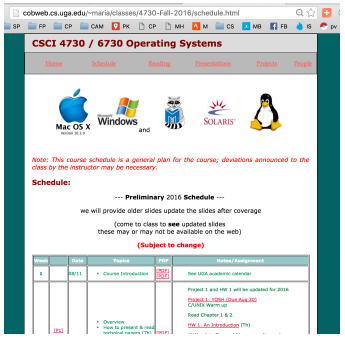
Logistics



- Who am I?
  - Office: Boyd 219C
- Class:
  - Boyd (blue)
  - Chemistry Room 455 (purple)
- Instructor contact:
  - maria.hybinette AT mac.com
  - Subject line: [x730 | 4730 | 6730]
- Office Hours: Tue after class (email to ensure)
  - And other times by e-mail appointment
- TA: TBD - check class web page for updates…

Communication Links

- http://www.cs.uga.edu/~maria/classes/x730-Spring-2018/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 / Class Forum: **Piazza**
- Turn-ins assignments on nike.
  - May be via submit command (old style) on nike.

Course Objectives

- Exposure to programming operating systems and its kernel
  - (programming) Practice on the concepts that make system 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.

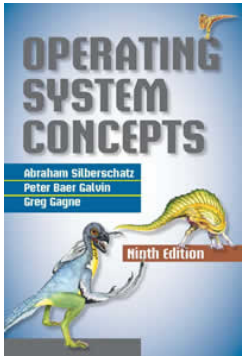
Course Concepts

Know and understand fundamental issues of operating systems

- **OS Structures**
  - Microkernels, Monolithic kernels, modular, Virtual Machines
- **Processes & Threads**
  - Communication:
    - Messages/RPC/IPC/RMI (possibly sockets)
  - Scheduling
  - Synchronization & Deadlock
- **Memory Managements & Virtual Memory**
  - Access memory how to make it more efficient
- **Storage Management**
  - File Systems
  - I/O System
- **Advanced Topics**
  - Security
  - Mobile computing

Class Structure

- **Read & Listen** – Textbook/Lecture:
  - Required: "Operating Systems Concepts", 9th Edition, Silberschatz, Galvin, Gagne (comes in e-format)
  - Recommended: "Modern Operating Systems", Tanenbaum
- **Practice**
  - 5-7 programming assignments
    - At least 5, more likely 6.
  - Technical paper presentations & summary.
    - 1-2 depending on number of students in class
    - 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



Grading (subject to change)

- Theory 40%
  - 2 Exams (10% each) + Final 15% + Quizzes 05% = 40%
- Practice 55%
  - 3-5 homework & weekly summaries (20%)
  - presentation (5%) &
  - programming assignments (30%)
- Participation 5%

- **100**% attendance will raise your final grade by 2%
  - Slides on web site vs. lecture may have some updates … that you may be interested in …
- We will use the College Board's convention to convert from percent grades to letter grades grades.
  - See class page for break down.

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**
  - **Project may be checked via in-class demo.**
- You are **encouraged** to ask questions of one another, and to respond to other student's questions (and especially on the email list - Piazza)
  - Part of your grade to be an active participant.
- 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
  - 1 for undergraduate students ('teams' of 1 or 2 depending on size of class).
  - 2 graduate students.
- 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 – 10-15 minutes long (about 10-15 slides)
- Core topics, research projects (e.g., clouds), project (C programming) oriented topics, or possibly : 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:
  1. What is the problem that the authors are trying to solve? [why is it important]
  2. What is their approach and how is it original and innovative? [compare against contemporary approaches]
  3. What are the assumptions/limitations?
     - Strength & weaknesses
  4. How is the approach evaluated:
     - What are the results/impact of paper (Why is this paper important, relevant)?
  5. 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.
- We will talk more about this next week.

Example Presentation Topics

- OS History (Unix/Linux/Windows)
- OS Structures
  - Micro/Monolithic Kernels
- Cloud Computing
- Mobile Computing
- Real Time OS
- Concurrency/Synchronization

Select a paper / from class repository / conference

Direction of the projects (in C) for class

Tentative projects (these may change)

1. **Multi-Process and Process Communication**
   - Example: Convert a single process program into a multiple process program
2. **Multi-Thread Programming**
   - Example: Programming a multi-threaded web server, however you my use html helper functions (i.e., you are not programming the IPC, or html protocol per se).
3. Investigate different **Schedule Protocols**:
4. **Synchronization:**
   - Conditional variables/Programming with Semaphores
5. Evaluate **Virtual Memory allocation** policies
6. Programming a **File System Simulator** (if time permits)

Prerequisites.

- Some experience in C or C++ programming (coding, testing and debugging).
- Not be afraid to program in C.
- Motivation

```c
#include <stdio.h>

struct arecord
{
  int i;
  float PI;
  char A;
};

int main()
{
  struct arecord ptr_one, *ptr_two;

  ptr_one.i = 10;
  ptr_one.PI = 3.14;
  ptr_one.A = 'a';
  ptr_two = &ptr_one;

  printf("First value: %d\n", ptr_two->i);
  printf("Second value: %f\n", ptr_two->PI);

  /* how do you print ptr_one ? */

  return 0;
}
```


Prerequisites: Know the some of the basics and the language

- **Basics**: What does a program really do when it run?
  - Fetches and instruction from memory, decodes it, then executes the instruction
- **The language**: Call procedures/functions, return from procedures/functions, access devices.

Prerequisites/Background

We will program in C, it is **expected** that you are a **motivated** 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

- Course Designers: Myself and Prof. Kyu H. Lee
- Tidbits & Material are drawn from several other 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)
- **Constructive Students Feedback**
- Wikipedia

Homework 1

- Get used to submission processes/and a C warm-up program:
  - Due Thursday

Quiz/HW: C - Practice

Please turn in on note book paper (1-3): Please tell us:

1. Name, major, year?
2. What computer hardware do you own (include smart phones if you own one)?
3. List the Operating Systems (OS) that are familiar to you (e.g. the OS on your laptop/the OS on your phone)?
4. Warm-up: (continue at home as part of HW1) 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);
```