CPS 510: Advanced Operating Systems  
Fall 2020

Class Meetings
Prerecorded lectures (approximately 90 minutes per week) on Panopto  
Tu 3:30 – 4:45 Ghost class (optional except by announcement; instructor available)  
Th 3:30 – 4:45 Weekly live paper discussion

Instructors
Michael Hewner  
Jeff Chase  
Danyang Zhuo  

Website
https://www2.cs.duke.edu/courses/fall20/compsci510/

This is advanced class in operating systems intended for graduate students and advanced undergraduates. An operating system is software that controls some programmable platform for sharing resources and data. All operating systems must deal with core issues of protection, resource management, program environment and execution, coordination, and reliable state storage and recovery. This course emphasizes advanced topics building on introductory material, with a focus on modern cloud/server environments. For example: we study container environments and the extended kernel abstractions they are based on; virtual machine hypervisors to extend basic material on protected kernels and virtual memory; memory models and scalable synchronization to extend basic concurrency control; enclaves as a secure variant of the classic process model; mandatory information flow control to illustrate advanced access control; various security topics; and file system designs focusing on performance, scale, safety, and extensibility.

Topics. A list of specific topics and related reading is available on the course web. We publish slides and other materials in a course git repository: see the course web. In Fall 2020, the CPS 510 is synchronized and interleaved week-by-week with an introductory course (CPS 310) that runs concurrently in the same time slot on MWF. These courses share a git repository and Panopto folder. You will encounter CPS 310 material: you are encouraged to review it as needed, but it is not required. We also share some labs for the first half of the semester.

Preparation. You should be familiar with undergraduate-level computer architecture/OS and consider yourself a strong student and a good programmer. We will ask you to program in C and in Java.

Tools. Know the basics of the Unix command interface. We will use the git revision control system and your project code will be hosted in private repositories on github. No public repositories please! (We have an ongoing problem with Duke students publishing their code for prospective employers, which requires us to hunt them down. Please please don’t do it.) The projects (labs) require an Ubuntu Linux environment, e.g., with Docker Desktop for your Windows or Mac machine. Labs are autograded by the AG350 service. Follow the directions on the course web.

Readings. There is no required textbook. We will use the Web-available text, Operating Systems in Three Easy Pieces (OSTEP) as introductory material. There will be weekly assigned papers.

What to do. This semester we have no exams, but we add weekly paper discussions or other activity in collaborative Zoom activity sessions. Every week we have: lectures (pre-recorded), a Sakai quiz on the week’s material (W), a synchronous activity session (Th), a reading reflection on Sakai (F), and a lab/project, which is due via AG350 on the following Monday. Please contact us if you cannot attend the activity.

Projects / labs. We have weekly projects for the first 10 weeks. Many are shared with the introductory class. They usually take 3-10 hours to complete, depending on the particulars and student background. Get
help from an instructor or TA if you’re stuck! There’s no point spending 5 hours debugging one issue. For the last 4 weeks we have an independent/team project—a larger freeform project allowing you to explore an area of OS on your own (within some guidelines).

**Grading.** Grades are based on a point score weighted as in the table. The weeks are equally weighted.

<table>
<thead>
<tr>
<th>Grading Category</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecture Quizzes (W 11:55 PM)</td>
<td>15%</td>
</tr>
<tr>
<td>Activity/Paper Reflections (F 11:55 PM)</td>
<td>15%</td>
</tr>
<tr>
<td>Labs/Projects (M 11:55 PM)</td>
<td>50%</td>
</tr>
<tr>
<td>Independent Final Project</td>
<td>20%</td>
</tr>
</tbody>
</table>

*Lecture Quizzes and Reflections.* Lecture quizzes are short quizzes design to ensure you watched the weekly lectures. For reflections, you answer questions on Sakai with your thoughts (<1 page). We will drop the lowest grade of both these categories, because we know that sometimes there are unexpected internet blackouts or other issues. Please finish these before the deadline and save your dropped grade for really unexpected problems.

**Late work.** Graded work has deadlines. To keep things fair no extensions can be granted to individuals or groups. We may occasionally grant extensions to the entire class, or grant “free passes” for late days that all groups may “spend” as they choose. Late work receives a penalty of at least 20%, or more, depending on circumstances. It is much better to do the work and hand it in late than to receive a zero on any assignment.

**Assistance.** We will provide online assistance through Piazza: see the course web. Please post your questions there. Piazza etiquette: check for answers before posting, keep posts clear and tight and include all relevant information, check your civility, don’t reveal too much in public posts. Also, you will find us to be generally welcoming—email or chat us on Teams for a live consultation if needed. Unfortunately, there are no TAs for this class.

**Policy on collaboration for CPS 510.** Collaboration on lab/project work is acceptable. We encourage social contact in these days of COVID. In particular, you are encouraged to share your knowledge, your understanding of the assignment, and your experience with “sticking points”. However, sharing of code across groups is not allowable. Any work you turn in must be your own, and you may be called upon to explain (alone) your choices and approaches in more detail. You may incorporate public software into your assigned lab work and course project to a reasonable extent, but not so much as to undermine the educational purpose and spirit of the project. You must acknowledge any sources of your words, ideas, and software when they are not your own, and you must disclose (in advance, in a README.txt file in your submitted code, without any specific request) any sources you consulted. Unacknowledged use of another’s code is cheating. Receiving assistance from any external source on quizzes or exams is cheating.

In particular, it is forbidden to incorporate code from students who took the course in a previous semester, or from students at other schools using similar projects. Please do not place your code in open repositories where they can tempt other students. If you discover one please let us know about it.

For practical reasons, we choose to treat code sharing among groups in the same semester as a public health problem rather than an offense under the university’s academic dishonesty policy, to the extent that the assistance is acknowledged. Your intent is to learn. Our intent is to help you learn and to avoid creating a climate of secrecy and fear around ordinary assistance or use of example code from documentation or other common sources. But do not share code across groups. All students should understand that we have software that flags copied code with a high degree of certainty and precision. (The tools do not differentiate the makers from the takers.) Sharing that is “over the line” may result in a downward grade adjustment, according to our discretion.