CPS 110: Undergraduate Operating Systems
Spring 2009

1 Basic Information

Dr. Landon Cox, D304 LSRC, lpcox@cs.duke.edu

Teaching Assistant
- Ryan Scudellari, scudella@cs.duke.edu

Web page: http://www.cs.duke.edu/courses/spring09/cps110
Newsgroup: http://courses.duke.edu
Note: **Use the newsgroup for technical questions regarding projects or lecture material**. Or see us in office hours.

Avoid emailing us directly for technical questions; it will only delay a response since we are likely to ask you to post your question to the newsgroup. We plan to check and respond to the group regularly. Other students may also be able to respond to your technical questions.

Office hours are posted on the course home page.

2 Course Overview

CPS 110 is an introductory course in operating systems at the advanced undergraduate or beginning graduate level. The objective of the course is threefold: to demystify the interactions between the software you have written in courses like CPS6 and CPS108 and the hardware you studied in CPS104, to familiarize you with the issues involved in the design and implementation of modern operating systems, and to explain the more general systems principles that inform the design of the Internet and other distributed systems.

The concepts in this course are not limited to any particular operating system or hardware platform. We will discuss examples that are drawn from historically significant and modern operating systems including Unix, Windows, Mach, and the various generations of Mac OS. We will cover topics such as processes and threads, concurrency and synchronization, CPU scheduling, virtual memory management, virtual machines, communication in distributed systems, file systems, and security.

To help you understand operating systems, you will implement several projects that form or depend on much of the core functionality in modern operating systems. These projects will give you practical exposure to topics such as threads, virtual memory management, virtual machines, client-server systems, and file systems. We will also assign homework to help you gauge your understanding of the material as the course progresses.

3 Prerequisites

Students must have obtained a grade of C or better in CPS 100 and CPS 104. Students are expected to understand computer architecture and data structures, to have extensive programming and debugging experience, and to be familiar with UNIX. All of the programming in this course will be in C++. The reason for this is that in order to truly understand how software and hardware interact through the operating system you have to dirty your hands with raw bytes. Languages like Java and C# intentionally abstract things like address pointers, memory management, and memory heaps away and, in most cases, such abstraction is a wonderful idea. Unfortunately, when trying to understand how an operating system works, it is a terrible idea, since the OS is where the software rubber meets the hardware road.

For some of you this will be the first time you have programmed in a messy, bug-inducing type-unsafe, garbage-uncollected language. That's ok; the first few projects are intended to help you become more comfortable writing code in C/C++. Students with questions about whether they have sufficient preparation for this course should speak with the instructors as soon as possible.

4 Course Materials

The course will primarily rely on lecture notes and projects as a way to learn the material. The **suggested** text for the course is *Modern Operating Systems* by Tanenbaum. Copies of Tannenbaum are easy to find on Amazon or other on-line sources. Students are required to read the home page and newsgroup frequently to stay current. Several helpful links to relevant programming material are posted on the web page.

5 Course Projects

Four projects will be assigned during the term. The first project is very simple and is only meant to help you get your feet wet with C++. **The other projects require a substantial time commitment on your part.**

5.1 Group Policies

All projects in this course are group projects, for which you must form a group of 2-3 students. Members of a group need not be in the same discussion section. To declare a group's membership, send e-mail to lpcox@cs.duke.edu with the group members' names and **computer science login IDs** (*not* your acpub ID or NetID---see Section 9). The group declaration deadline is Friday, January 9. After this date, we will combine remaining students into groups of 2 or 3. It is very important to choose your partners carefully. You should discuss topics such as prior experience, course background, goals for this course, workload and schedule for this semester, and preferred project management and work style. Before agreeing to form a group, make sure you are able to agree to several blocks of time during the week to meet to discuss or carry out the project.

Exams will almost certainly contain questions related to projects. Students are expected to participate wholly in their group to the benefit of the entire group. All group members should be familiar with all aspects of the project, irrespective of their role on the project. We expect all group members to contribute their fair share. **Each group member may be asked to submit an evaluation of contributions of other group members after each project**. Members who contribute less than their share may receive a lower grade on the project; non-contributing members will receive a zero. In case of disputes regarding contribution, a TA or professor may interview group members. Students may be "fired" from a group by all the remaining members or "quit" a group. The procedure for this is as follows:

(1) documented "gentle warning" of risk of firing or intention to quit in e-mail, with cc to all group members and to lpcox@cs.duke.edu, with cause and/or specific work required to remain in group

(2) allow at least 72 hours

(3) after 72 hours, if the problem is not resolved, send documented statement of firing or quitting in an email, with cc to all group members and to lpcox@cs.duke.edu.

Note that a fired student or a student that quits a group may need to work alone if unable to join another group. A student working alone will be graded on the same curve as the other groups. Managing group dynamics and effectively using each group member's time and talents can be as difficult as solving the project. We are happy to offer advice on how to handle these issues.

Be open and candid with your group about any potential problems early on so that your group can plan around such problems and not fall behind. A sure way to make your group upset at you is not finishing your part of the work at an agreed-upon deadline *and* not informing them about the problems early enough for them to help.

5.2 Turning in Projects

You will be submitting your projects electronically by running a program called submit110. Projects are due at 6:00 pm on the due date. To account for short-term unexpected events like computer crashes, submission problems, and clock skew, we will allow 6 hours of slack and accept projects until exactly 11:59 pm. Sometimes unexpected events make it difficult to get a project in on time. For this reason, each group will have a total of 3 late days to be used for projects throughout the semester. **These late days should only be used to deal with unexpected problems such as illness, hospitalization, or personal and family emergencies**. They should not be used simply to start later on a project or because you are having difficulty completing the project. Projects received after the due date (assuming that you have no late days left) will receive a zero, even if it is just one second late. **DON'T CUT IT TOO CLOSE—one zero can have a drastic effect on your grade!** Try to save one or two late days for the last project.

Weekend days are counted in the same way as weekdays (e.g. if the project deadline is Friday and you turn it in Sunday, that's two days late). Each submission by any group member counts as a submission by all members of the group (i.e. you can't use one member's 3 late days on one project and another member's 3 late days on another project; all late submissions get charged to all group members). Individual/group **extension requests (other than the use of free late days) will not be granted**, even for computer problems, loss of data, network being down, family emergencies, etc. If a family/personal emergency causes you to miss significant number of days in the semester, please come and see us to decide the best course of action. In general, you can avoid most problems by starting the projects early and keeping backup files. If you are having trouble understanding the material or designing a program, please come to office hours for help right away. In most cases, with reasonable cooperation and good faith on your part, your project group members may be able to make up some of the deficit without needing an extension if you have extenuating circumstances.

5.3 Doing your Own Project

All projects in this course are to be done by your own group. Violation will result in initiation of the formal procedures of the University Honor Councils, as appropriate. At the same time, we encourage students to help each other learn the course material. As in most courses, there is a boundary separating these two situations. You may give or receive help on any of the concepts covered in lecture or discussion and on the specifics of C++ syntax. You are allowed to consult with other students in the current class to help you understand the project specification (i.e. the problem definition). However, you may not collaborate **in any way** when constructing your solution—the solution to the project must be generated by your group working alone.

You are not allowed to work out the programming details of the problems with anyone outside your group. You are not allowed to look at or in any way derive advantage from the existence of project specifications or solutions prepared in prior years (e.g. programs written by former students, solutions provided by instructors, project handouts).

If you have any questions as to what constitutes unacceptable collaboration, please talk to the instructor right away. The instructors may use automated tools and carry out manual inspection of the submitted code to detect similarities with submitted solutions. You are expected to exercise reasonable precautions in protecting your own work. Don't let other students borrow your account or computer, don't leave your program in a publicly accessible directory, do not share your solutions via a web site, and take care when discarding printouts. On the other hand, if you find such printouts left mistakenly at printers or code of

other students available publicly, inform the person to whom the code belongs (or us, if possible) and do not use the content in your project.

5.4 Tips for Success in the Projects

The most common reason for not doing well on the projects is not starting them early enough. You will be given sufficient time to complete each project. However, if you wait until the last minute to start, you may not be able to finish. Plan to do some work on a project every day. Also plan to have it finished a few days ahead of the due date; many unexpected problems arise during programming, especially in the testing phase. The computing sites can become quite crowded as deadlines approach, making it difficult to get a computer. Plan for these things to happen.

Your lack of starting early is not an excuse for turning in your project late, even if some unfortunate situations arise such as having your computer crash. Late days should only be used to deal with unexpected problems such as computer crashes, illness, submission problems, or deadline conflicts. They should not be used simply to start later on a project or because you are having difficulty completing the project. Try to save one or two late days for the last project.

The most common reason for spending too much time on a project is hacking before thinking. Resist the urge to start banging out code as your first step; you are likely to code yourself into a corner. Sit down together with your project partners and plan out the architecture of your solution. Expect to revise this architecture several times before settling on a plan. Pay particular attention to the project descriptions, and generate a list of behaviors the specification requires of your solution. Design your project with independently and incrementally testable subsystems rather than save all testing for the end. Assign one project member as the Testing Czar; that member's job is to see how they can break the team's solution.

The second most common reason for spending too much time on a project is hunt-and-peck debugging: trying things at random, just to see if they work. If you find yourself in this position, step away from the keyboard and think about what is happening, or come see us in office hours. There are many sources of help from which you can draw. Most questions can be submitted to the TAs, professors, and your fellow classmates via the course newsgroup. These will typically be answered within the day, often more quickly during working hours. We ask that you **do not** pose questions via email, as you will not receive a response as quickly, and the rest of the students in the course do not benefit from the answer. If you have a question that is inappropriate for the newsgroup, (for example, you have a specific question about your own code, and cannot post the question publicly), please see a member of the course staff in person during office hours rather than send email.

Many computing sites have consultants available. They are fine sources of help with questions regarding the computers and installed software (such as Unix, news, and the C++ compiler). However, they are not likely to be able to help you with questions about computer programming, the C++ language, or specific errors in your program.

6 Homeworks

We will assign short homework sets for most weeks. The homework questions will be covered in the discussion section. Discussions with other students in the class on homework sets is allowed, as long as you document whom you collaborated with. However, you must create and turn in your own document. Homework may or may not be graded and will be used as part of the participation portion of your grade.

7 Exams

There will be two exams during the semester: a midterm on Tuesday, February 23rd during class, and a final exam on Wednesday, April 29 from 7-10 pm. You are expected to take both exams at the scheduled times. If you miss an exam, you will receive a zero for it, unless a documented medical or personal emergency causes you to miss it. If you anticipate conflicts with the exam time, talk to the instructor at least **1 month** before the exam date. The exam dates are given at the beginning of the term so you can avoid scheduling job interviews or other commitments on exam days. Outside commitments are not considered a valid reason for missing an exam.

8 Grading Policy

Final grades will be based on the total points earned on the projects and exams. Grades will be assigned on a curve. Factors such as class participation and ability to work well in a team may be used to adjust your final grade if it falls near a borderline. The tentative point breakdown is as follows:

- Projects 35%
- Midterm 30%
- Final 30%
- Participation 5%

Incompletes will generally **not** be given. In accordance with university policy, doing poorly in a course is not a valid reason for an incomplete. If you are having problems in the course, your best bet is to come talk to the instructor as soon as you are able to.

9 Computing Environment

You may use any workstation that runs Linux and g++, including a virtual workstation (i.e. VMware virtual machine). If you do not have an account with the CS department, we will need to obtain one for you. Please send an email to lpcox@cs.duke.edu with your preferred or existing login ID by Thursday, August 30. You can then login into your account by sshing to linux.cs.duke.edu. You should use this account for all auto-grader communication.