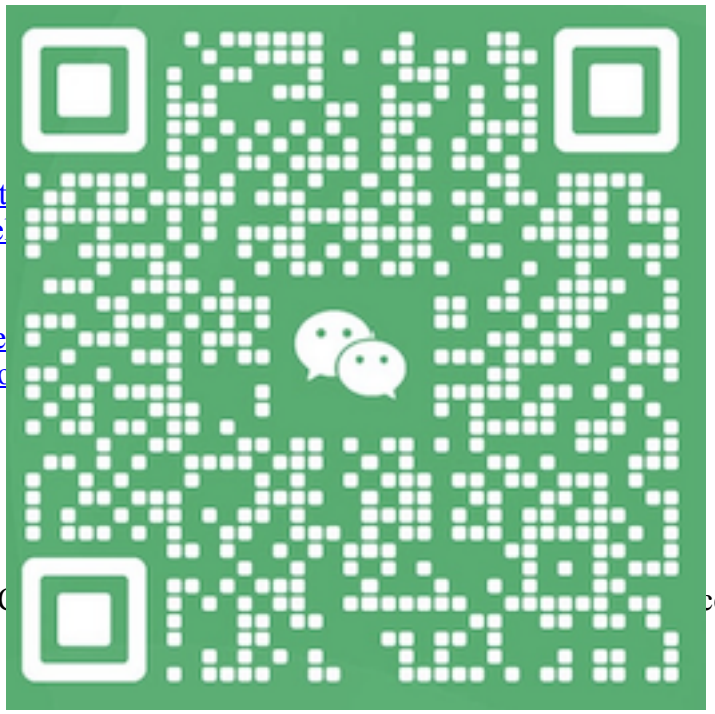
COMP 530: Operating Systems

Syllabus and Policies

1. Bulletin Description
2. General Course Info
3. Instructor Info
4. Teaching Assistant(s)
5. Textbooks and Resources
6. Course Description
7. Target Audience
8. Prerequisites
9. Goals and Key Learning Objectives
10. Course Requirements
    1. Lecture Recordings
11. Key Dates
12. Grading
    1. Labs
    2. Examinations
    3. Homework
13. Course Policies
    1. Lateness
    2. Honor Code Stat
    3. Cheating Yoursel
14. Attendance
15. Title IX Resources
16. Accessibility Resource
17. Counseling and Psych
18. Syllabus Changes
19. Acknowledgements



Bulletin Description

Types of operating systems. C cesses, devices. Scheduling, protection. Case

General Course Info

Term:	Fall 2023
Department:	COMP
Course Number:	530
Section Number:	001
Time:	MW, 3:35-4:50 PM
Location:	SN 014
Website:	http://www.cs.unc.edu/~porter/courses/comp530/f23

Instructor Info

Name:	Dr. Donald Porter
Office:	Fred Brooks 344
Email:	porter at cs dot unc dot edu
Phone:	919-590-6044
Web:	http://www.cs.unc.edu/~porter
Office Hours:	Scheduled on Course Care, or by appointment.

Teaching Assistants

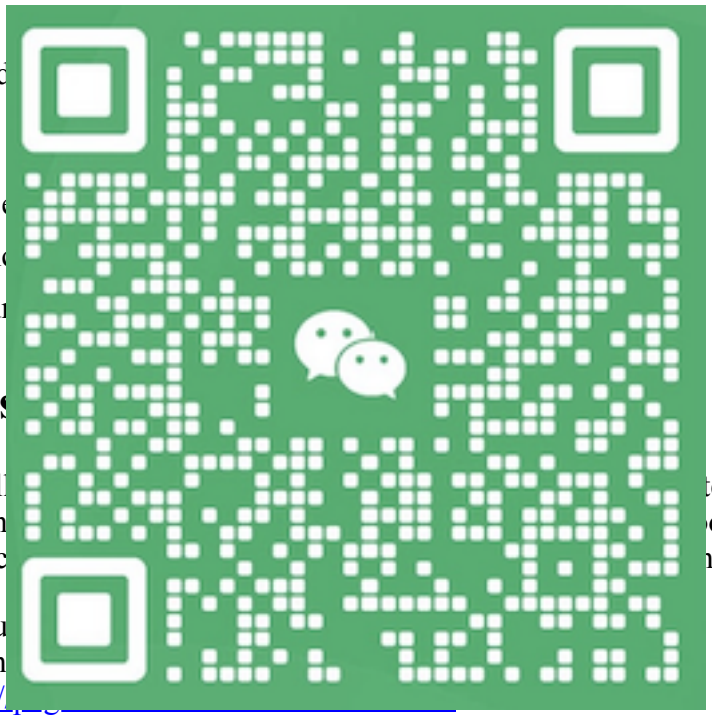
All office hours will be scheduled on Course Care, or by appointment.

Name	Email
Eric Schneider	eric at cs dot unc dot edu

Learning Assistants

All office hours will be sched

Name	
David Karash	dwkarash at
Felipe Yanaga	yanaga at unc
Amin Zamani	azamani at u



Textbooks and Res

Required Textbook: We wil ted on the course schedule. The readings will n ok with additional required and optional resourc e course schedule.

- Remzi Arpaci-Dusseau
  Operating Systems: Th
  Free online here: http:/

Most of the course materials will be on the course website. We will also use Campuswire for class discussions, and Sakai only for posting assignment grades.

A number of helpful references for the labs are available on the References page.

Optional Textbooks: The following books are useful references for this course and OS kernel programming in general. These books are available from amazon, and several are available for free online via the UNC library or course reserves on canvas.

Other recommended operating system textbooks and references (in no way are these required purchases for the course):

- Thomas Anderson and Michael Dahlin
  Operating Systems: Principles and Practice, 2nd Edition

Recursive books (August 21, 2014),
ISBN: 0985673524

- Marshall Kirk McKusick, George V. Neville-Neil
  The Design and Implementation of the FreeBSD Operating System
  Pearson Education, 2004.
  ISBN: 0201702452

- Uresh Vahalia
  Unix Internals: The New Frontiers
  Prentice Hall, 1996.

- Andrew S. Tanenbaum
  Modern Operating Systems
  Prentice Hall, 1992.

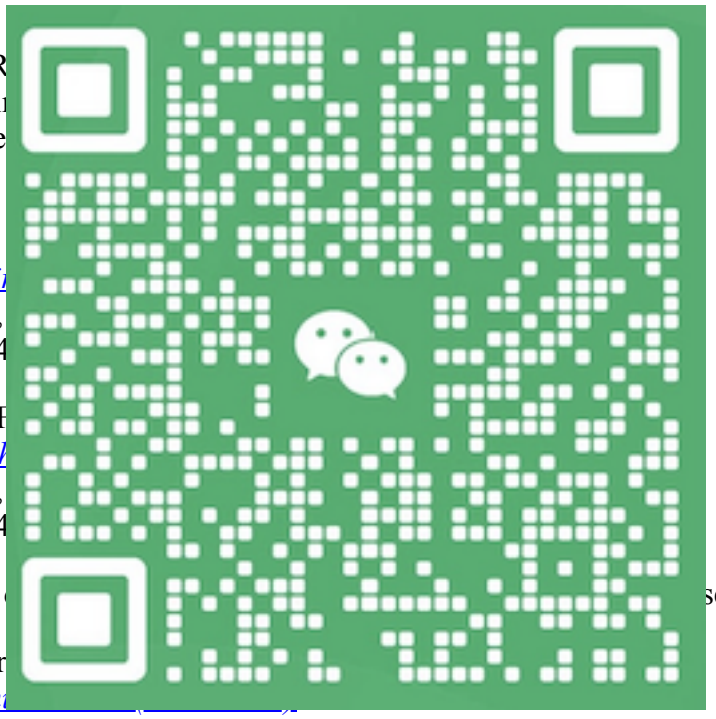
If you need help with C or Unix, I recommend these texts:

- Brian Keringhan and Dennis Ritchie
  The C Programming Language (2nd ed., ANSI version)
  Prentice-Hall Software Series, June 1988.
  ISBN: 0131103628

- Brian Keringhan and R
  The UNIX Programmi
  Prentice-Hall Software
  ISBN: 013937681X

- Æleen Frisch
  Essential System Admi
  O'Reilly & Associates,
  ISBN: 978-0-596-0034

- Ellen Siever, Stephen F
  Linux in a Nutshell, 6t
  O'Reilly & Associates,
  ISBN: 978-0-596-1544



If you are interested in more se references:

- Daniel P. Bovet & Mar
  Understanding the Lin
  O'Reilly & Associates, Novemeber 2005.
  ISBN: 0596005652
  Note: Be sure NOT to get the older editions of this book, which covered Linux 2.2 (1st ed.) and 2.4 (2nd ed.).

- Jonathan Corbet; Alessandro Rubini; Greg Kroah-Hartman
  Linux Device Drivers (3rd edition)
  O'Reilly & Associates, February 2005.
  ISBN-13: 978-0-596-00590-0
  This book is has a more accessible introduction to compiling your own kernel and writing your own module than Bovet and Cesati, as it is intended to be a practical guide to writing device drivers.

- Robert Love
  Linux Kernel Development (3nd Edition)

Addison-Wesley Professional, 2010.
ISBN: 0672329468

- Christian Benvenuti
  Understanding Linux Network Internals-->*Understanding Linux Network Internals*
  O'Reilly Media; 1 edition (December 1, 2005)
  ISBN: 0596002556

Course Description

An operating system is an essential part of almost all computer systems. In fact, your cell phone, your car, and most consumer appliances that contain a processor have an operating system inside. Your car probably has at least 10 different operating systems lying about.

At a high-level, the principles of operating systems include topics such as how to securely and fairly share resources among multiple applications; how to design abstractions for hardware resources that balance ease of programming with expressive power; and techniques for coordinating concurrent access to a resource.

More specifically, this course will primarily study general purpose, time-shared operating systems. In this context the operating system is the software system that provides the interface between users, their applications, and the underlying hardware. The purpose of this course is to introduce some of the fundamental concepts in the d e:

- Processes and inter-pro
- CPU scheduling
- Memory management
- Secondary storage man
- File systems
- Deadlock detection and
- Security and authentica

More philosophically, a comp w any part of the computer as "magic," but sho figure it out. For instance, when one types a c nd software events that lead to the command returnin

Thus, an important part of the will developing assignments on a Linux syste programming.

Target Audience

The course is geared towards advanced undergraduate computer science majors and first year graduate students in computer science. Operating systems is a classic topic in a computer science curriculum as the problems of resource allocation, management of concurrency, and file storage have always been present in some form in nearly all computing environments. You can't understand how a computer operates until you understand what an operating system is, how it functions, and how it is organized.

Prerequisites

The prerequisites for this class are:

- COMP 210 (Data Structures) or COMP 410 (Data Structures - in the old sequence)
- COMP 311 (Computer Organization) or COMP 411 (Computer Organization - in the old sequence)

These courses are necessary background. If you have not completed COMP 210 and 311 you will be dropped from the class.

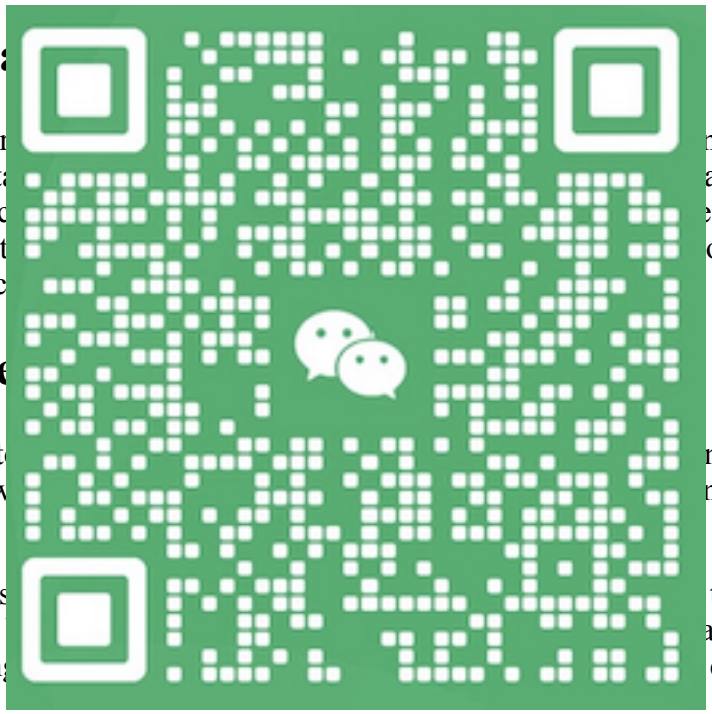
If you've taken equivalent courses elsewhere, and they include *actual programming experience* in C/Unix, please speak to me first to get an approval to take this class. If you've never taken an introductory C/C++ course before, you may not take this class; in some cases, having industry experience in the same field is enough.

C Programming: You should already know the basics of programming and debugging in **C programming language**. This course will be taught in C, and you will complete substantial, user-level programming assignments in C. We expect that a typical student will become much more proficient in C over the course of the semester. However, because this is not a C course, the time spent in class on C will be minimal.
If you do not know C and would like to take the course, I would recommend reading "The C Programming Language" by Kernighan and Ritchie and working the exercises in the book **BEFORE THE FIRST DAY OF CLASS**. A dedicated student that is proficient in another language (e.g., Java) can probably accomplish this in a week or two.

You should already have basic exposure to Unix commands and the command line. You should know what commands such as ssh, gcc, make, man, ls, mkdir, vi/emacs, and gdb do, or be able to figure this out on your own (via google and friends). In general, we will not teach you how to use Unix (we'd like to spend the time teaching you about operating systems instead), although we may discuss particularly tricky commands.

If you do not fulfill the above requirements, you should very strongly consider postponing COMP 530 until you are more prepared.

Goals and Key Le



Students will learn how to wr ntly executing processes, and how to communicate dat ate system resources such as memory and processor cyc ct system performance. They will also learn how syst ocesses so as to provide the abstraction that each proc

Course Requireme

The course is taught in an int nned topics to discuss, students are expected to answ ndance is expected, but will not be recorded.

To help students review and s to be completed in a small group outside of class. a social media site for the course. The groups will chan complete approximately 8-10 worksheets.

Students are also expected to complete roughly 4 programming assignments. The programs will be written in the C programming language on a departmental Linux server and will each emphasize some aspect of operating system design and implementation.

Lecture Recording

Lectures will be recorded and be made available to the students in the class. These recordings are intended to help students review the material after attending lectures, and are not a substitute for attending lectures in person. Lecture attendance is still expected, to facilitate questions, announcements, and discussion. If lecture recording substantially harms attendance, it will be discontinued.

This is a best-effort service and should not replace lecture attendance. Student questions, chalkboard drawings, and other materials may not record properly. Moreover, my experience has been that a few lectures

are lost each semester for unforeseen technical difficulties (e.g., the recording space fills up mid-lecture, a file gets corrupted).

Key Dates

There will be two in-class exams: Weds 10/4, and Mon 11/13, held during class time.

There will be a final exam at 4pm on Thursday, Dec 14.

Please mark your calendars now. **If you have a conflict with the midterm, tell the instructor during the first two weeks of class, and we will schedule a makeup for a time before the exam is given to the rest of the class.**

Grading

The final grade will be determined as follows: The raw scores obtained by all students on each assignment and exam will be standardized for that particular assignment or exam either (at my discretion) by converting them to percentile scores, or else by applying a linear transformation to map the scores to a standard [0, 100] scale. A weighted sum of the resulting standardized scores will then be formed (with weights as shown below) to obtain a composite score for each student.

- Labs (45%)
- Exam 1 (15%)
- Exam 2 (15%)
- Final examination (25%

Finally, the composite scores [illegible] my choice to determine the cutoffs for each grade cat[illegible] of composite scores, information derived from late[illegible] to contribute to this decision.

In assigning final grades, I of[illegible] course of a semester. However, I will not entertain [illegible]

So that you can get an idea of[illegible] rough percentile information when I hand bac[illegible] omputed until after all grade changes and correction[illegible]

The exams are closed book a[illegible] programming questions.

Re-grading: we will handle regrade requests on gradescope. It's highly recommended that you take some time to review your entire assignment before requesting a regrade. **We reserve the right to regrade the entire assignment or exam, not just the question raised in the regrade request. Your grade can be improved or harmed by regrading.**

Extra credit: Some labs may include optional challenge problems, which may be completed for extra credit. Please indicate if you do these in your lab's challenge.txt file. The instructor may also assign bonus work in class at my discretion. Any extra credit points accrued by any student will be used as follows. The final course grade will be assigned as a letter grade which excludes all extra credit points. Then I will apply a subjective method to determine how much value to assign to extra credit points. Extra credit points can only be used to raise your final course letter grade. In other words, **you are not obligated to do any of the extra credit work**, and you can still get an A in this course.
A note of caution: in the past, some students have spent too much time working challenge problems and gotten behind on core assignments; note that the relative value of extra credit is small compared to the main course assignments.