CSE 3100 Systems Programming – Spring 2021

Lecture

Mon/Wed 11:15am–12:05pm, AUST 108 or by web-based videoconferencing.
Videoconferencing access links will be posted for each class meeting on the Moodle site.
All class activities through Jan. 29 and following the Spring break are held by videoconferencing only.

Lab

Sections 001L/101L: Th 12:20PM - 2:10PM, by videoconferencing
Sections 002L/102L: Th 10:10AM - 12:00PM, by videoconferencing
Sections 003L/103L: Th 2:30PM - 4:20PM, by videoconferencing
Sections 011L/111L: Fr 8:00AM - 9:50AM, by videoconferencing
Sections 012L/112L: Fr 10:00AM - 11:50AM, by videoconferencing
Sections 013L/113L: Fr 12:00PM - 1:50PM, by videoconferencing
Section 014L: Th 12:20PM - 2:10PM, ITE 138 (by videoconferencing Jan. 21 & 28 and April 22)

Instructor

Ion Mandoiu
ion@engr.uconn.edu
Office Hours:
Tu/Th 12pm-1pm
ITE 261 or by videoconferencing
Piazza Live Q&A:
Wed 7-8pm

Teaching Assistants

Austin Abate
austin.abate@uconn.edu
Office Hours:
Wed 1:20-2:20pm

Bazz Coleman
john.b.coleman@uconn.edu
Office Hours:
Tu 2-3pm

Md Fahim
md.fahim@uconn.edu
Office Hours:
N/A

Jordan Force
jordan.force@uconn.edu
Office Hours:
Th, 2:30-3:30pm
ITE lobby

Jack Grossman
jack.grossman@uconn.edu
Office Hours:
Wed 2-3pm

Dominic Martire
dominic.martire@uconn.edu
Office Hours:
Wed 4-5pm

Finn Navin
thomas.navin@uconn.edu
Office Hours:
Tu 10-11am

Samuel Oslovich
samuel.oslovich@uconn.edu
Office Hours:
Mon 2-3pm

Yiming Zhang
yiming.zhang.cse@uconn.edu
Office Hours:
N/A

Catalog Description

Introduction to system-level programming with an emphasis on C programming, process management, and small scale concurrency with multi-threaded programming. Special attention will be devoted to proficiency with memory management and debugging facilities both in a sequential and parallel setting.
Prerequisite: CSE 2050 or 2100

Course objectives

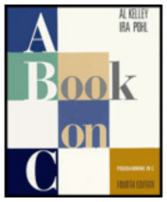
- Understand the C language and, in particular, memory management and pointers.
- Understand inter-process concurrency primitives such as pipes, sockets and virtual memory mapping and how to use them to build concurrent client-server applications.
- Understand intra-process concurrency primitives, in particular, POSIX threads and synchronization primitives such as mutexes and condition variables needed to write multi-threaded applications

Learning Outcomes

By the end of the semester, students should be able to:

- Write, compile, debug, and execute C programs that make use of memory management functions.
- Write, compile, debug, and execute C programs that use files and I/O on UNIX.
- Write, compile, debug, and execute C programs that create, manage and terminate processes and threads on UNIX.
- Write, compile, debug, and execute C programs that use UNIX synchronization primitives.
- Write distributed applications that communicate across a network.

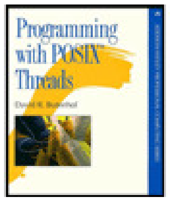
Required Textbook



Al Kelley and Ira Pohl
A Book on C, 4th Edition
Addison-Wesley, ISBN-13: 978-0201183993
Book website including source code for all examples:
https://users.soe.ucsc.edu/~pohl/abc4.html

Optional Textbooks

Brian W. Kernighan and Dennis M. Ritchie
The C Programming Language, 2nd Edition
Prentice Hall, ISBN-13: 978-0131103627



David R. Butenhof
Programming with POSIX Threads, 1st Edition
Addison-Wesley, ISBN-13: 978-0201633924

Online platforms

Moodle. We will use a course website hosted using Moodle at https://edx.engr.uconn.edu/. Please use the Moodle site to connect to class meetings and office hours, and to access interactive videos, assignments, grades, and other course materials. The Moodle site also integrates video-calling spaces hosted on gather.town to facilitate study in groups.
Piazza. For electronic class discussions we will be using Piazza, which can be accessed from Moodle (the first access will ask you to confirm joining our Piazza class if you have not done it already). You are strongly encouraged to ask class-related questions and communicate with other students, the instructors, and the TAs via Piazza rather than e-mail. Please observe basic etiquette by keeping your messages polite, concise, and on-topic. Before posting new messages do take a look at the postings that are already there–it is possible that your question has already been answered. Appropriate questions include general questions about the material covered in class and clarifications on the assignments. Keep in mind that the collaboration policy is in effect and you must not post extensive code fragments in public messages. For questions that are specific to your work use direct messages to the instructors or the TAs.
Mimir. Labs, homework assignments, and exams will all involve writing C code and must be submitted electronically via the Mimir platform. The first access to Mimir from Moodle will ask you to link an existing Mimir account (or create one if needed). This allows Mimir to transfer your grades to Moodle and enables access to Mimir from Moodle without additional authentication. The first lab will review the Mimir IDE and the submission process as needed.

Grading

Asynchronous course content (graded interactive videos and quizzes) will be posted weekly on Moodle. It is essential that you review the asynchronous content and complete associated quizzes prior to each class/lab meeting to ensure you are prepared to participate in class discussions. In addition to interactive videos and quizzes, grading will be based on weekly labs, homework assignments, and three exams. Labs are short C programming exercises designed to give you hands-on practice with common programming tools and an opportunity to apply the concepts covered in lectures. Homework assignments will require you to write more complex C programs, often building on a provided code base. The exams will consist of programming tasks similar to those in the labs and homework assignments.

Grade breakdown

Interactive videos & quizzes	10%
Labs	10%
Homework assignments	20%
Three Exams	20% each

The lowest homework assignment score and lowest lab score will be dropped from the overall grade calculation.

Late policy

Labs and homework assignments will typically be due at midnight on the specified date. To ensure timely feedback, including the release of sample solutions, late submissions will not be accepted.

Collaboration policy

Unless otherwise indicated, all lab and homework assignments must be completed individually. All programs and documents you hand-in must be your own work. You may discuss course related topics with others, but must not share code or quizz answers. Reasonable use of published materials (including web resources) is allowed, but all sources must be explicitly acknowledged in your submissions. Violations will be reviewed and sanctioned according to the University Policy on Academic Integrity. An example of unreasonable use is submitting copied solutions with minor changes like renaming variables. If you need additional clarifications regarding the collaboration policy, please contact the instructor.

Students with disabilities:

If you have a documented disability for which you are or may be requesting an accommodation, please contact the Center for Students with Disabilities by the end of the third week of the semester to ensure that any accommodations you need can be implemented in a timely fashion.

Tentative Schedule

Dates	Lecture/Lab topics
Jan 20 Jan 21/22	Course introduction; C overview Lab1: Mimir IDE, basic Linux commands
Jan 25 & 27 Jan 28/29	Expressions & types, control flow Lab2: Building C projects
Feb 1 & 3 Feb 4/5	Functions; structures and arrays Lab3: Debugging
Feb 8 & 10 Feb 11/12	Pointers and memory management Lab4: `valgrind`
Feb 15 & 17 Feb 18/19	Standard library and I/O Lab5: profiling
Feb 22 & 24 Feb 25/26	Miscellaneous C topics Exam1
March 1 & 3 March 4/5	Processes and pipes Lab6: `fork()`
March 8 & 10 March 11/12	Signals and intro to sockets Lab7: pipes
March 15 & 17 March 18/19	Client-server communication using sockets Lab8: sockets
March 22 & 24 March 25/26	Sockets Selection Exam2
March 29 & 31 April 1/2	Intro to threads, thread management Lab9: thread management
April 5 & 7 April 8/9	Thread synchronization: mutexes, spinlocks, and condition variables Lab10: thread synchronization
April 19 & 21 April 21/22	Thread synchronization: read-write locks, barriers, and semaphores Lab11: debugging and profiling threads
April 26 & 28	Miscellaneous multithreading topics: false sharing, threads scheduling and priority inversion, GPU computing