EECS 398 System Design in C++
Winter 2019 Syllabus

Instructor Nicole Hamilton
nham@umich.edu
https://web.eecs.umich.edu/~nham/
C: 425-765-9574

Lectures Dow 1014
Tuesdays and Thursdays, 3:00 pm to 5:00 pm
Lectures will be recorded but attendance is strongly advised.

Office hours Beyster 2649
Monday through Thursday, 5:00 pm to 6:00 pm

Prerequisites EECS 280 and 281

Classification Depending on your major:
CSE and CE: MDE or CS Flex Tech
CS-LSA: Capstone or CS Flex Tech
EE: Technical elective
DS: Advanced Technical or Flex Tech elective

Overview

This is a course in how to tackle a large system programming project in C++. You'll work on a team of 6 to write a complete multithreaded internet search engine from scratch.

My objective is to offer students an opportunity to work on a significant *relatable* design project (something you can explain to your family and to recruiters) early in your undergraduate careers. I want the experience to be like working on the startup dev team for a new product (like the one I had on the startup team for what became the Bing engine.)

Sixty percent of your grade will be on the project and will be competitive, based mostly on your team's performance compared to the others and on your individual contribution to your team. Thirty percent of your grade will be on the exams. Ten percent is homework.

This will be the second time for this course. The engines submitted last time ranged in size from about 4.2 to just over 14 KLOC; most were about 6 KLOC. (Yes, a little smaller than you might have guessed.) Sounds easy (maybe) but you will deal with a lot more ambiguity than in most classes, in part because the course is still so raw and under construction.

Learning objectives

Here are main learning objectives:

1. The experience of working on the startup dev team for a significant project in C++, building a complete working internet search engine from scratch.
2. The knowledge, skills and ability to work with others to research, define, estimate, plan and carry out the design of such a large, complex software project, especially, one you don't already know how to do.
3. An understanding of how a large system is decomposed into separable parts with interfaces exposing a limited surface area and how they talk to each other.
4. An understanding of and ability to use the low-level operating system facilities for files, process and thread creation, synchronization, IPC and sockets.
5. An appreciation of software as art and of the software design process as a fundamentally creative activity where you make choices.
6. The ability to communicate your results by demonstrating your product, producing a final report and presenting your work to the class and invited guests in a professional manner.

System design

System design projects always seem to have some defining characteristics.

1. There's an important domain-specific part that asks you to learn something new about an interesting problem you've never seen before, in this case, how a search engine works.
2. There's a need to invent a solution, an architecture, breaking the problem down into lots of moving parts, with lots of data structures and algorithms.
3. It's usually "close to the metal" with lots of low-level OS calls everywhere and the need to define file and interchange formats, perform handshakes, share resources using locks, deal seriously with error recovery, etc.
4. They're usually team efforts because they're too big to do any other way.
5. You build the whole thing, often from scratch, and at the end, you get to see it work and it feels good.

Every new design project always has a new domain-specific part, which keeps a career in system design interesting, but the rest of the skills are the same and they're things most people learn by doing.

What I like about a search engine as a case study in system design, and the reason the course is called system design, not "how to build a search engine", is that it's really compact fun project that hits on every bit of what, to me, system design is all about.

The project

You will self-select into teams of 6 to design and build a complete working internet search engine, assigning your own roles. Choose your teammates wisely. It's helpful if your team has a mix of skills, including one or two who've had relevant electives like 482 or 445. But to be successful and to be a great teammate, it's more about being willing to dive in and do your part to help your team build stuff than about how many electives you've taken and how much you know going in. If you've taken 280 and 281, I will try to cover the rest of what you need to know to build any part.

These are the basic pieces you will need to build.

1. HTML parser.
2. Crawler.
3. Index.
4. Constraint solver.
5. Query language.
6. Ranker.
7. Front end.

In lectures, we'll walk through the problem posed by each piece and how it might be solved. You will also find the text helpful in explaining how a search engine works. A lot of the pieces need to talk to the operating system, so we'll go through a lot of toy examples in C++ on both Windows and Linux showing, e.g., how to memory-map a file, create a thread or open a socket.

But you will have to make many of your own decisions as a group about what your architecture should look like, what's minimally needed and what's nice to have, how each piece will work, how the pieces will talk and how you'll get it all done on time.

All your work will be in C++ 11.0 and all of it must be yours. All your code must adhere to the style sheet I'll provide. I'm still struggling with how to set policy but I prefer you not use a lot of STL outside of test cases. I'd like you do your own problem-solving, inventing your own data structures and templates. I think it will raise your awareness of lower-level design and performance issues.

Your grade on the project will depend in roughly equal measure on your individual and team performance. Your team performance will be determined on a competitive basis by ranking your engine's overall performance, features, index size and quality against the engines created by the other teams. I will also consider your team's ability to communicate your results through your presentation, demonstration and final report.

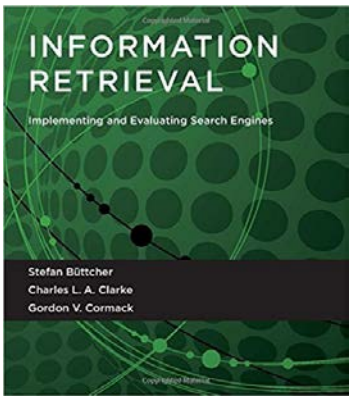
Your individual performance on the project will be based on your ability to work with others and contribute to your team's success. I'll primarily consider the number of lines of code you write, the complexity of the tasks it performs, its performance, the creativity it displays, and the overall elegance. I will also consider how you've contributed to the atmosphere, enthusiasm, creativity, progress, inclusiveness, camaraderie, health and overall success of your team.

Homework

I anticipate perhaps a half-dozen homework assignments at two or three-week intervals, some individual and some to be worked as a team.

Texts

There will be only one required text. What I like about *Information Retrieval* is that it describes a lot of the pieces (except for ranking!) in the same way we thought about it at Microsoft.



Information Retrieval: Implementing and Evaluating Search Engines
Reprint edition (February 12, 2016)
Stefan Büttcher, Charles L.A. Clarke, Gordon V. Cormack
The MIT Press
ISBN 978-0262528870

Grading

Here's a rough idea of the weighting I expect to apply.

Group project and other activities	
Group performance	30%
Individual contribution	30%
Homework	10%
Midterm	15%
Final	15%

I grade on the curve, scaling the results by deciding what I think was an A and what I think was a C and then interpolating in between. I expect but do not promise that most students will fall between 2.8 and 4.0 with median around 3.2 or a little higher.

Late policy

Late submissions will not be accepted.

Late withdrawals

I intend to deny virtually all requests for withdrawals once the teams are formed and underway.

Course revisions

This is only the second time for this course so of course you should expect I'm still learning how to make it better and that I may make reasonable revisions to content, assignments, exams or grading as may be appropriate.

All the work must be your own

You may certainly compare notes with other students and other teams and of course I understand that you may do research using Google. But absolutely everything you turn in to me must be your own work. On your exams, you will be required to copy the Honor Pledge in your own handwriting and sign your name to it.

> *I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the Honor Code.*

I report everything.

Disability

Access and Accommodations: Your experience in this class is important to me and to the University of Michigan, where it is our policy and practice to create inclusive and accessible learning environments consistent with federal and state law.

If you experience barriers based on a temporary or permanent disability (including, but not limited to mental health, attention-related, learning, vision, hearing, physical or health impacts), please seek a meeting with the Services for Students with Disabilities (SSD) office at 734-763-3000 or https://ssd.umich.edu to help us determine appropriate academic accommodations. SSD typically recommends accommodations through a Verified Individualized Services and Accommodations (VISA) form. Any information you provide is private and confidential and will be treated as such.

If you have already established accommodations with SSD, please tell me what they are so I can be sure to provide them.

List of topics

Here is a rough outline of the topics we'll discuss, interwoven over the semester. We obviously won't dive into everything to the same depth.

1. Introduction to the course. Expectations, grading, coding style requirements. Brief overview of the project and of how a search engine works.
2. The role of ideas and creativity in software design. Software as art. Beautiful code, efficiency, elegance. The style guide we'll follow.
3. The software design process.
    a. The initial vision. Something has become possible.
    b. The typical dev organization, devs, testers, program managers, leads and managers.
    c. Research, reverse engineering, brain-storming.
    d. Problem-choosing, deciding what's in and what's not.
    e. Decomposition into an architecture, a design strategy and a working definition of the interfaces.
    f. The product spec.
    g. Planning and estimating, KLOCs and other metrics, milestones.
    h. Problem-solving, coding and testing.
    i. Code freeze, triage, release.
4. Search engine basics.
    a. Brief history of information retrieval and web search.
    b. The concept of relevance.
    c. Dynamic and static rank.
    d. Text formats and code sets, Unicode and UTF-8.
    e. Simple HTML, title, heading, body text, links and anchor text.
    f. Tokenization, stop words, stemming.
    g. Crawling and the frontier.
    h. The inverted word index and the constraint solver.
    i. Query compiling, BNF grammars, TDRD parsers.
    j. Ranking and learning techniques.
    k. Snippets.
    l. Spam, link farms, traps, objectionable content.
5. OS facilities.
    a. Processes and threads.
    b. The file system.
    c. Pipes.
    d. Producer-consumer relationships.
    e. Shared memory.
    f. Memory-mapped files.
    g. Locks, critical sections, race conditions.