

Course intro

CSE 2421 – Systems I

Introduction to Low-level Programming and Computer Organization


Neil Kirby

Available on Zoom or Baker Systems 418D

Kirby.249@osu.edu

If you are on the waitlist for this class

- ▶ Business as usual

- ~~1. The sections are all full, you only get in if someone drops~~
 2. Be sure to sign in on the attendance sheet
 3. CSE Majors & Data Analytics Majors (along with a few others) get priority
 - ~~4. We are at 125% enrollment – there isn't any more room~~
 5. I prefer people who show up for my class to get into my class
- 

Lecture Style

- *Interactive – you are coming with me on this journey*
- *I will call for questions as I go – I mean it*
- *If you don't understand, feel free to ask questions*
- *I will ask questions in class*
- *If we go virtual:*
 - *Chat is the vehicle for questions that you want to ask*
 - *Chat is the vehicle for answers to questions that I ask. I'm totally good with 50 people all typing "C is call by value" into chat. Typing those answers helps you learn*
 - *If you are really fast, wait on some of them to let others be first*

Lecture Style

- *I am enthusiastic*
 - *That's polite for saying sometimes I tend to rip through slide decks; if you all have it the first time, I won't waste your time*
 - *It is OK to ask me to slow down*
- *I have a BSEE & MSCIS and my first programming assignment for Bell Labs was C and Assembler*
 - *Expect to hear real-world illustrations*
 - *I'll do my best to keep them relevant*

My Graders

8 Graders for 2 sections of 2421 and 1 of 3902:

CSE 2421:

Lennon Anderson (anderson.3901)

Chris Barlas (barlas.3)

Stephanie Dietrich (dietrich.167)

Ben Elleman (elleman.28)

Julia Rank (rank.31)

Kyle Rosenberg (rosenberg.1278)

CSE 3902:

Alex Contreras (contreras.84)

Tyler Bramley (bramley.6)

Course Textbook(s)

Computer Systems: A Programmer's Perspective, 3rd Edition, by Randall E. Bryant and David R. O'Hallaron, Carnegie Mellon University (Required)

- ▶ We are using the electronic version that includes MyLab and Mastering
- ▶ There is an assignment in the assignment are on this
- ▶ The electronic aids should prove quite helpful in the later 2/3 of the semester

Course Textbook(s)

Pointers on C, by Kenneth Reek 1997 (Recommended, not required)


- Amazon prices (8/25/20)
 - ~~Rent~~
 - Buy used \$100
 - Buy new \$157
- The paperback (international) version is less expensive, reviews say the quality of the book and printing is really bad, but they don't complain about content

If you chose not to purchase this book (since it was recommended but not required), keep a note of the title and author. If you end up in a position where C is a part of your daily responsibilities, buy it. Highly recommended. There is a copy of this text in the Engineering Library.

Objectives – superficial

- ▶ Learn to program in C
- ▶ Learn to program in x86 assembler

Objectives – deeper

- ▶ Be able to design and code larger and more complex programs in C
 - ▶ Be able to handle low level concepts such as pointers and direct memory access
 - ▶ Be able to integrate with libraries and code written by others, and quite possibly to read and understand code written by others
 - ▶ Improve coding style to be coherent and concise
 - ▶ To be aware of performance impacts
 - ▶ To be able to read your code for what it says, not what you want it to mean
- 

Course syllabus

- ▶ The syllabus is on Piazza
- ▶ If you have questions or need help, please ask, either during class, by posting on Piazza, or by stopping to see me or one of the graders during their office hours (or by making an appointment).
- ▶ My office hours and the graders names/office hours are posted on Piazza on the staff tab
- ▶ I do not regularly check email on Carmen; use Piazza or email to kirby.249@osu.edu
- ▶ All slides will be posted to Carmen before class
- ▶ Grades will be posted to Carmen.
- ▶ Strongly consider bringing your iPad (or better) to class to take notes on and to do in-class assignments

Grading

Your grade for this class will be determined by:


Description	Percentage of Grade
Programming Assignments (6)	10%
Homework Assignments (*)	10%
In-Class Assignments (*)	10%
Midterm	30%
Final Exam – must pass	40%

- * a mix of simple 1–point and complex ~10 point assignments
- ▶ You must pass the final exam in order to pass the course.

About In-class Assignments

- ▶ Any lecture day could have an in-class reflection
- ▶ Everyone works in groups but each student submits their own assignment
- ▶ Lowest ones are dropped
- ▶ If you need to miss class, arrange in advance
- ▶ *You are allowed to ask me if what you have is correct before you turn it in*
- ▶ Anything you see here is fair game for the exams

Interactions

- ▶ Homework and in-class assignments should be done in groups
 - ▶ Bullying and personal attacks are not permitted.
 - ▶ A 1 or 2 full letter grade penalty will be levied for such behavior beyond any university sanctions. (B+ becomes D+)
 - ▶ If you think a grader is treating you unfairly, bring the facts to me. (This is different than disagreeing on how the lab was graded – in that case you go to the grader first and only afterwards to me)
- 

Check your calendar

- ▶ Let me know in advance if you have events such as conferences this semester
- ▶ Let me know in advance if you have religious holidays that could have an impact
- ▶ Military? Marching Band? On the team? Let me know in advance
- ▶ I teach 2 sections of 2421 we will have combined evening exams. *Go look at the tentative exam dates and email me with any conflicts at least 2 weeks in advance (ten seconds from now is not too soon).*

Principle Goal

- ▶ The principle goal of education is to create men and women who are capable of doing new things, not simply repeating what other generations have done

– Jean Piaget

- ▶ In other words, men and women who can “think” rather than those who can “memorize answers”

– Janis Jones



Personal Goal

- ▶ My goal is: students who pass my sections are better prepared than students from other sections.

(Somebody has to be the best – why not you?)

(And hey! Think about grad school!)


In this class I will ask you to raise your game.



Topics for the course

- ▶ The course material is divided into two large parts:
 - Part 1: The C programming language
Programming & debugging in C along with the tools to do so
 - **Part 2:** Introduction to computer systems and X86-64 assembly language
Programming & debugging in x86-64 along with the tools to do so
- ▶ Sprinkled liberally within the two sections:
 - Binary, octal, decimal and hexadecimal number representations
 - Performing mathematical and logical operations on the above
(e.g. addition, subtraction, XOR, AND, OR, L/R shifting)
- ▶ Anything else related to these topics that I can find that will make you better employees and better computer scientists

Likely Additional Topics

- ▶ ARM architecture
 - ▶ RISC vs CISC
 - ▶ Hardware
 - ▶ Performance
 - ▶ Etc.
- 

Topics for today

- ▶ How to setup the lab environment we'll be using for the remainder of the semester
- ▶ An overview of Java vs. C
 - After tomorrow, the “Java” word in this classroom will only refer to coffee
- ▶ A quick summary of the lab assignment that will be assigned tomorrow if we have time

Lab environment (for both parts of the course)

- ▶ You will do lab work on stdlinux servers in CSE.
- ▶ You can access stdlinux in two ways:
 - 1. Go to a CSE lab location: info on labs at <https://cse.osu.edu/computing-services/computing-labs/locations-hours>
 - 2. Remote access: <https://cse.osu.edu/computing-services/resources/remote-access>
- ▶ I strongly recommend that you use one of the remote access methods *and test it today!*
- ▶ I will try to help you with remote access issues, but I cannot make any guarantees. Your best bet is to seek help at the help desk <https://ets.osu.edu/>

More on remote access: Windows

- ▶ Windows: You need a secure shell client to run on your Windows machine.
- ▶ Two options:
 - PuTTY
 - FastX2
 - See the remote access information at the CSE link on the preceding slide to get instructions. Read the sections labeled UNIX, PuTTY, and FastX2
 - PuTTY is far more reliable than FastX – every semester, somebody gets screwed by FastX and networking issues; if the network works at all, PuTTY works.

More on remote access: Linux/Mac

- ▶ Linux/Mac: Open a terminal, and using your user name, type:
 - `ssh -X username@stdlinux.cse.ohio-state.edu`
 - After a connection to the server is established, you will be prompted for your password.
 - After entering your password and hitting return, you should get a linux prompt showing your home directory.
 - If you don't know your CSE username or password, you need to go to 392 Baker Systems in person. Here's a link for them: <https://ets.osu.edu/>

MATE is not your friend

- ▶ Teach yourself how to work in a command line interface world
- ▶ UNIX commands are fair game on exams
- ▶ In the real world there are Linux/Unix environments that only have command line interface access

More on Labs: *Caution!*

- ▶ C Labs for this course must build on stdlinux using gcc with **particular** command line options
 - *Do not use IDEs– they will be more trouble than they are worth!*
 - A lot of students write a perfect lab in Windows with Visual Studio, etc., . . .
 - . . . then fail the lab because they can't get it to compile in a Linux environment
 - It's not worth the time to convert from one to the other, and you lose the benefits of learning to work in a Linux/Unix environment, which you may very well need to do later in your career.
 - Note: If you do things this way, any resulting problems are your responsibility.
 - A few students in previous classes chose not to heed this warning and ended up with no points on labs they put a lot of work in. ☹

VSCode can be used if you are careful

- ▶ See “VSCode_Documentation.pdf” in the General Resources section of piazza
- ▶ You are totally on your own here
- ▶ That documentation was provided by one of my former graders – be nice to my grading team because they do stuff like this that makes your life easier
- ▶ I don't use this because I type really fast and have a zillion vi commands committed to memory, not because of any failings in VSCode

Why OSU teaches Java first

- ▶ Excellent tools: IDE, CI, etc.
- ▶ Many 3rd party libraries
- ▶ Lots of documentation
- ▶ Platform independent
- ▶ Reasonable performance
- ▶ Well thought out exception handling
- ▶ Native Threads
- ▶ Easy to Learn
- ▶ Lots of available jobs
- ▶ A lot changed between 1972 and 1995
- ▶ Classes and the advantages of OOP.
- ▶ Memory management (garbage collection)
- ▶ Security (partly related to the absence of pointers)
- ▶ Larger libraries (no need to re-create the wheel)
- ▶ Ability to put something together more quickly that works and *does* something

Some motivation for Systems I course content

You might all be asking yourselves these questions:

- ▶ Why study C?? It's old technology, Java and C++ are what's being used now!
 - See the next few slides
- ▶ Why would ANYONE want to study assembler in this day and age?!?!?!?
 - Assembler language is still often used for performance-critical parts of a program
 - Interfacing with or fixing code for which you have no source code
 - Assembler code knowledge is indispensable when diagnosing malware (security jobs)
- ▶ I'm not an EE. I'm not planning to create circuit boards. Why do I need to know about processor architecture?
 - If you are going to program, knowledge of the hardware makes you a better programmer
- ▶ Are you a “one-trick pony”? Do you want to be? Where you be in 10 years? Is it where you will want to be?

Why study C?

TIOBE Index for August 2022

1. Python 15.42%
2. C 14.59%
3. Java 12.40%
4. C++ 10.17%
5. C# 5.59%
6. Visual Basic 4.99%

C has been in the top 3 for nearly 20 years

<https://www.tiobe.com/tiobe-index/>

Why Study C?

- ▶ C is universal;
- ▶ C runs directly on the hardware, rather than requiring an Interpreter which slows down the execution speed (i.e. C runs faster than just about anything except assembly language);
- ▶ It has a very powerful and varied repertoire of operators;
- ▶ Elegant syntax
- ▶ Ready access to the hardware when needed;
- ▶ Compile once, run anytime;
- ▶ Preferred language for implementing other programming languages and compilers;
- ▶ Lots of C code still out there. Make yourself more marketable.

Why Study C?

- ▶ “High Level” languages provides everything the programmer might want to do already provided for in the language (e.g. Java)
- ▶ C is considered a “Middle Level” programming language. It has the capability to access the system’s low level functions. It provides building blocks that you will need to produce the results your are being asked to provide.
- ▶ “Low Level” languages (assembler) provide nothing but access to the machines basic instruction set.
- ▶ 3/7/2016 blog <https://www.pluralsight.com/blog/software-development/why-every-programmer-should-learn-c>

Not Java Time! Welcome to C!

- ▶ Going from Java to C is like going from an automatic transmission to a stick shift
 - Lower Level: much more is left for you to do (usually more efficiently)
 - Unsafe: you can set your computer on fire
 - C standard library: much smaller
 - Similar syntax: can both help and confuse
 - Not object oriented (it's functional): paradigm shift – this might be the hardest part
 - “This is how the world really works.”


People who program in C for a living say:

Java is to “paint by numbers”
as
C is to “oil paint on canvas”

(They may be right. You can tell me what you think at the end of the semester.)




So which is better, Java or C?

- ▶ A person with a more sophisticated view of technology (I hope that describes all of you 😊) would not ask this question, but rather, a rather different one:
 - What are the advantages of C, and what are its disadvantages, and likewise, what are the advantages of Java, and its disadvantages?
 - ▶ What's better? A hammer or a screwdriver?
 - Doesn't the answer depend upon what you're trying to accomplish?
 - Use the right tool for the job – last summer I learned how to use a pickaxe
 - ▶ Based on this assessment, we can make good choices about which language might be the best in a given situation.
- 

Java



Java is a good application for:

- ▶ Cell phones
 - ▶ desktop computers
 - ▶ Web servers
 - ▶ Application servers
 - ▶ Rapid development
- 


Advantages of C (and assembly language, later in the course)

- ▶ They do exactly what you tell them to do
- ▶ No built in memory management (memory leaks, OH MY!)
- ▶ C allows the use of pointers, which are powerful
 - #1: require care to use effectively
 - #2: require rigorously logical thinking to use correctly
- ▶ C is compiled to machine code (in several steps), so the compiled program runs directly on the hardware.
- ▶ Wonderfully, yet irritatingly, obedient – you type something incorrectly, and it has a way of compiling fine and just doing something you don't expect at run-time.
- ▶ No frills, no overhead, just raw power/speed
- ▶ This is what makes it challenging
- ▶ And this is what makes it fun!

C/assembly



C is a good application for:

- ▶ Performance Critical Applications and Resource Limited Systems
 - Games (along with C++)
 - codecs
 - IoT (the Internet of Things includes the Internet of **small** things)
 - ▶ Operating Systems (linux is mostly in C)
 - ▶ Compilers
 - ▶ Programs that operate hardware (e.g. robotics)
 - ▶ Networking
 - ▶ Just about anything where raw power and speed are required
- 

Other reasons to learn C in the context of the goals of this course

- ▶ Another purpose of this course is to get a systems level view of how computer programs are written and run.
- ▶ We want you to develop a better understanding of the systems software (compilers, assemblers, linkers) which plays a role in how programs actually run on the hardware.
- ▶ The way that C language programs are converted to machine language instructions which can actually run on hardware, provides a “deeper” view of the relationship between a high level language program and its execution on the hardware.
- ▶ When we learn assembly language in the latter part of the course, we will also develop a better understanding of the hardware itself, and how it executes a program.

Lab 1 assignment

- ▶ Purpose: to get acclimated to the stdlinux environment, and the process for submission of labs (submitting zip files to Carmen), first work with debugger
- ▶ Detailed description of the assignment will be posted tomorrow.
- ▶ Two Due Dates: Early due date (10% bonus)/Real due date
 - This will be a *feature* for *each* and every lab you have
- ▶ Generally, you should read and follow the instructions for the labs *carefully*.
- ▶ People sometimes make avoidable mistakes, and lose all the credit, or significant numbers of points because the directions for the lab (in all sections) are not followed. Following instructions is important when you have a job – and a customer, and it's important here, too. The graders and I are your customer for this class. We will not grant you relief from errors you make because you did not read, or did not follow, the instructions.