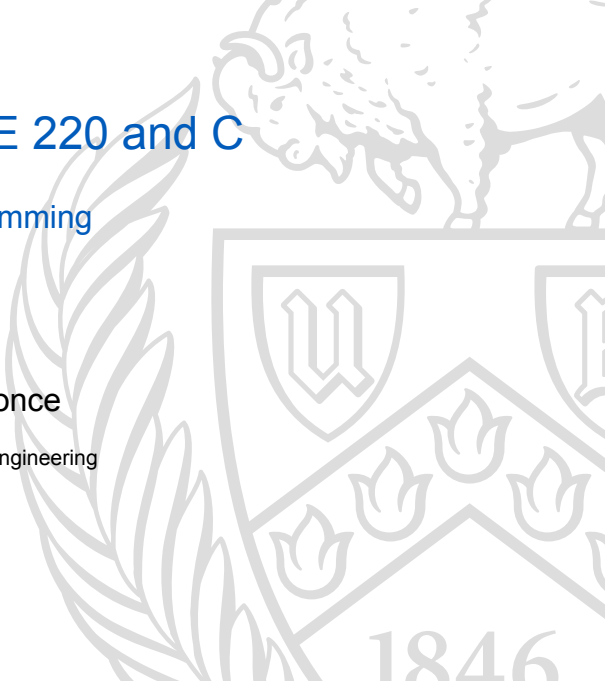
Introduction to CSE 220 and C

CSE 220: Systems Programming

Ethan Blanton & Carl Alphonce

Department of Computer Science and Engineering

University at Buffalo



Welcome to CSE 220

Contacting instructors:

Name:	Ethan Blanton	Carl Alphonce
Email:	eblanton@buffalo.edu	alphonce@buffalo.edu
Office:	Davis 334	Davis 343
Office Hours:	M 10:00–10:50	T 08:00–09:20
		W 10:00–10:50
	or by appointment	

The syllabus is on the course web page[1] and UBlearns.

So are these — and all other — slides!

[1] https://www.cse.buffalo.edu/~eblanton/course/cse220/

Systems Programming

This course is concerned with systems programming.

You will learn:

- More about the properties of computer systems
- How architecture affects programs
- How to effectively write efficient and correct programs
- The C programming language and POSIX API

Our goals are to:

- improve your intuitions about how computers work
- Practice solving real problems with programming

Programming in Context

Programming doesn't occur in a vacuum.

Computer systems have greatly influenced our:

- Programming languages
- Development tools
- Preferred algorithms

This course will help you understand that context.

Course Difficulty

This course is NOT a “weed-out” course!

It is, however, quite difficult.

It is difficult because:

- The material is precise and unforgiving.
- 2xx courses provide less guidance than 1xx courses.
- Success requires careful time management.

You will get out of this course what you put into it.

Expectation

For this course, we expect that you:

- Will be respectful to course staff and classmates
- Attend every lecture
- Attend every lab
- Adhere strictly to the academic integrity policy
- Will seek assistance early if necessary
- Meet prereqs; among other things:
  - Have some experience programming
  - Understand linked lists and object references

Most of all, behave as adults and strive to maximize your and your classmates' learning experience in this course.

Expectation

For this course, you can expect that all course staff:

- Will be respectful to you, your classmates, and course staff
- Will maintain a welcoming and productive learning environment
- Will support your learning of the course content
- Will help you become an independent learner; this means
  - not giving direct or complete answers to some questions
  - asking you questions to show you how to problem-solve
  - referring you to existing resources which address your problems

Most of all, we will behave as adults and strive to maximize your and your classmates' learning experience in this course.

Attendance

Lecture attendance is mandatory.

- We will not repeat lectures.
- Labs will not repeat lectures.
- You are expected to catch up on your own if you skip.

Lab attendance is mandatory.

- Lab attendance will be tracked.
- You will practice what you learn in class.
- You will be frequently tested on your understanding.

Attendance is monitored and required to pass this course.

Succeeding in CSE 220

Students who fail or resign CSE 220 often:

- Miss lectures or labs
- Start assignments at the last minute
- Don't visit office hours (early enough)
- Don't ask their questions on Piazza
- Don't commit to git
- Don't submit to Autograder early
- Cheat

Please address any problems early!

Readings

Most lectures will have both required and optional readings.

Readings will appear at the end of the slides.

You must read the required readings.
…Even if we do not mention them in class.

You may wish to read the optional readings:

- to expand your understanding of related topics
- to help you understand the required material

Assistance

Our primary forum for assistance will be Piazza.

You should have been added to our course Piazza.

Please consult existing postings before asking a question!

- However, if you're not sure your question is answered, ask!
- If you wish to include code, solutions, *etc.*, send a private message to the instructors!

We will post important course announcements and materials to Piazza.

Programming Projects

A significant portion of your course grade will be projects.

- These are individual projects.
- Projects will be written in C.

Projects must run on `emon.cse.buffalo.edu`.

Teaching yourself C[2] is a learning objective in this course![3]

[2] With our help!

[3] Remember, you have already learned Python and Java!

Course Time-sharing System

We provide you with a time-sharing system to do your development.

This is a shared computer that everyone will use.

It provides a uniform environment for all students.

Its name is `emon.cse.buffalo.edu`.

You will access it via a tool called Xpra.

Learning some Unix usage is a learning objective in this course!

Project Assistance

Your TAs will be your primary source of help for projects.

To get the most out of your TAs, do:

- try the obvious things first,
- create minimal examples to show problems, and
- consult the documentation.

To avoid wasting TA time and failing to get help, don't:

- ask for help before you've tried to understand the problem
- start at the last minute.

Editors

We don't care what editor you use, but it must be a programmer's editor, and be capable of:

- Syntax highlighting
- Automatic indentation
- Brace/parenthesis/*etc.* matching
- Extensibility

Neither instructors nor the TAs will help you if you are not using an appropriate environment when you seek help!

Our personal recommendation is Emacs.
Emacs is installed on the time-sharing system.

Today's Assignments

Immediately:

- Read the Syllabus.
- Watch the academic integrity video on Panopto.

By **Beginning of lab this week**:

- Create a GitHub account if you don't already have one.
- Make sure you can log into the time-sharing system.
- Watch the lab video and read the handout.

By **Wednesday next week**:

- Take the academic integrity quiz.

Lab

We have labs this week!

Go to your lab.

Grading

Passing this course requires five extra conditions:

- Completion of the AI quiz with perfect score
- Completion of Lab 01 with perfect score
- At least 60% weighted exam average
- At least 60% lab exam average
- Attendance to at least 30 lectures

Failing any of these points means an F in 220.

Your course grade will be calculated according to the Syllabus.

Why C?

There are dozens of programming languages. Why C?

C is “high level” — but not very.

- C provides functions, structured programming, complex data types, and many other powerful abstractions
- …yet it also exposes many architectural details

Most operating system kernels are written in C.

Many runtimes and virtual machines are written in C.

C influences many other languages.

Effective C

Effective C programming requires that you master the machine.

You must be aware of its architecture and details of operation.

We will be using C in Linux on x86-64.

The dialect of C that we will use is C99.[4]

The compiler that we will use is `gcc`.

[4] K&R describes ANSI C (C89), but we will discuss the differences when important.

CSE 220 and C

That said, CSE 220 is not (only) about learning C.

Lecture teaches concepts, and you will implement them in C.

We will not cover all details of C syntax.

We will cover key ideas and particularly important syntax.

Learning C will teach you to learn new languages.

You should consult:

- The C Programming Language (K&R)
- Unix man pages
- Given code

On Precision

This course will attempt to be precise, but must simplify some things.

Usually this is because the details:

- are unnecessarily confusing, or
- require knowledge you are not expected to have.

If something here conflicts with the standard or the compiler, the standard or compiler wins.

We will try to mark imprecise statements with a pilcrow: ¶

The Processor and Memory

The C language exposes a particular machine model.

Data is stored in memory at accessible addresses.¶

The CPU manipulates data stored in memory.

Program code is executed as a series of instructions:

- Also stored in memory
- Though possibly not accessible (as data)

A Dedicated Computer

Most modern, multi-tasking OSes (including Unix) provide a particular model.

That model is that each process has its own dedicated machine.

Each process appears to have:

- A dedicated CPU
- Private, dedicated memory
- Private input and output facilities

That isn't strictly true, but it is approximated by the OS.

The OS provides mechanisms to share resources in this model.

Programs as Instructions

C programs[5] are translated into machine instructions.

The computer executes these instructions in order.¶

Instructions are things like:

- Add two numbers together
- Compare a number to zero
- Store a number to a location in memory

As we will see, it's all bits.

[5] Indeed, all programs!

(Some practice TopHat questions, as time permits.)

Developing Hello World

"Hello World" is a classic first program when learning a language.

We will develop a Hello World together.

Summary

- C is a high level language used in systems programming.
- Architectural details are important in C.
- The C/POSIX model is:
  - A dedicated machine for each program
  - Sequential execution of program instructions
  - Data is stored in accessible, addressed memory
- We explored some trivial C programs.

Remember your required readings!

Next Time …

- More about types
- Variable declaration and usage
- C Strings
- Looping

References I

Required Readings

[1] *Course Syllabus*. https://www.cse.buffalo.edu/~eblanton/course/cse220-2023-2f/materials/syllabus.pdf.

[2] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Second Edition. Introduction, Chapter 1. Prentice Hall, 1988.

License

To retrieve a copy of this material, or related materials, see `https://www.cse.buffalo.edu/~eblanton/`.