CSE 13S: Computer Systems and **C** Programming

Prof. Darrell Long

Fall 2019

C programming, command line, shell programming, editors, debuggers, source code control, and other tools. Basic computer systems, algorithm design and development, data types, program structures. Develops understanding of process model, compile-link-execute build cycle, language-machine interface, memory, and data representation. Students cannot receive credit for both CSE 13S and CSE 13E. Course is 7 *units* $(5+2)$ *with integrated laboratory.*

1 Introduction

For this course, you are expected to have some programming experience. This usually takes the form of assembly language in CSE 12/L and you may also have taken CSE 20 or CSE 30, and so may have some experience programming in Python. Please be aware that programming in **C** is more like programming in assembly language than it is in Python: you are exposed to the machine, and it will not try to prevent you from making mistakes.

You are expected to have a certain level of mathematical maturity, corresponding approximately to Algebra 2, although it would be best if you had just a little Calculus since we will use derivatives in lecture.

You will learn the rudiments of some fundamental data structures: arrays, pointers, bit vectors, stacks, queues, hash tables, and simple trees. You will learn how to create abstract data types in a language that does not support object-orientation. You will learn about dynamic memory management, and how to twiddle bits.

You will be learning to use a number of tools that are common to UNIX (and have been ported to other operating systems as well). There are more tools than we can possibly cover during the quarter, so you will be expected to learn how to find the right tool for the job. You are encouraged to share with your fellows the tools that you find—but not the code that you write to use them.

You will encounter debuggers, static and dynamic code analyzers. You will see how code is translated from **C** to assembly language to machine code and then executed as a process.

You will learn computer systems concepts such as processes, memory, files, permissions, and the like. Ideally, at the end of this course you will be comfortable operating in a UNIX environment.

2 Reading

One of the most important lessions during your time in college is *learning how to learn on your own.* To a certain extent, that means that you need to learn to be an autodidact. You do this by reading; reading books, manuals, and published articles. These are not skills that you can learn by only watching videos on the Internet.

2.1 Required

All of your professors learned to program in **C** using the classic text by Kernighan & Ritchie. Here in the United States the cost is $59 on `Amazon.com`, but it is only $3.73 on `Amazon.in`. Take from that what you will regarding the pricing policies of the publishers.

- Brian W. Kernighan and Dennis M. Ritchie. *The C programming language.* 2006.

It is important for you to understand that reading *The C programming language* is not a suggestion, it is a requirement. You are expected to read it. And to then to read it again. This is not a onerous task: the book is not long, you can read it in an afternoon. That is the beauty of this book, and of the **C** language.

2.2 Recommended

On the campus network, access to many electronic books is *free of charge*. This is an enormous boon to you, and you are strongly encouraged to take advantage of it.

If you follow this link: `https://proquest-safaribooksonline-com.oca.ucsc.edu` you will find *many* electronic books that will be useful in this course (and others). We will suggest some readings in these books: please, do not ignore those suggestions.

Here are some of the books that we will refer to during the course of the quarter:

- J. Loeliger & M. McCullough. *Version Control with Git.*
- C. Newham & B. Rosenblatt. *Learning the bash Shell.*
- A. Robbins, E. Lamb & L. Hannah, *Learning the vi and Vim Editors.*
- A. Robbins, E. Siever, S. Figgins & R. Love. *Linux in a Nutshell.*
- T. Crawford & P. Prinz. *C in a Nutshell.*
- R. Mecklenburg. *Managing Projects with GNU Make.*

3 Grading

In this course, you will have a final examination that counts for 30% of your grade. The final examination is comprehensive, and you can expect it to cover all of the material that you have learned in the course. You will be allowed to bring one sheet of notes to the examination. Electronics of any kind, excluding medical devices, are disallowed.

We will be experimenting with weekly quizzes this term. The quizzes will be opened on Friday, close on Saturday. You will have an hour to complete them. It should go without saying—but obviously it does not—that you must complete these quizzes on your own, with no help from any other person (including virtual persons on the Internet).

These quizzes are in replacement of a midterm examination and count for 20% of your grade. This saves us from losing a day to the midterm, but if for some reason these quizzes do not work out then we will fall back to the position of having a midterm.

The programming assignments are worth 50% of your grade. You are strongly encouraged to attend the laboratory sections where specifics of each assignment will be discussed. But do not expect the TA or the tutors to write your code for you: they can help you to find bugs, and offer suggestions, but the ideas in the code must come from you, and the typing of the code must be done by you.

The points are allocated to assignments based on difficulty. For example, assignment 0, which is trivial, has very few points allocated to it. The assignment where you implement several sorting algorithms and do some comparison and analysis has correspondingly more points associated with it.

We will release assignments one week early. We are doing this in hopes of encouraging you to start early. It would be a mistake to not take advantage of this policy.

4 Programming

The most important component of this course will be programming in the **C** programming language. You will also learn to program using `make`, `bash`, and many other tools.

The programs start out trivial—a junior-level student should be able to complete the first assignment in about 10 minutes—but rapidly increase in difficulty. The first six assignments are allocated one week; the last two assignments are allocated approximately two weeks each.

Here is a list of the programming assignments for the quarter:

0. "Hello World!"
1. Temperature conversion
2. *Left, Right, Center*
3. Simple numerical library
4. *Towers of Hanoi*
5. Bit Vectors
6. Sorting
7. Hashing & Linked Lists
8. Data Compression

You will be using a lot of tools, some which you may have never seen before. You will, for example, be using the *command line*:

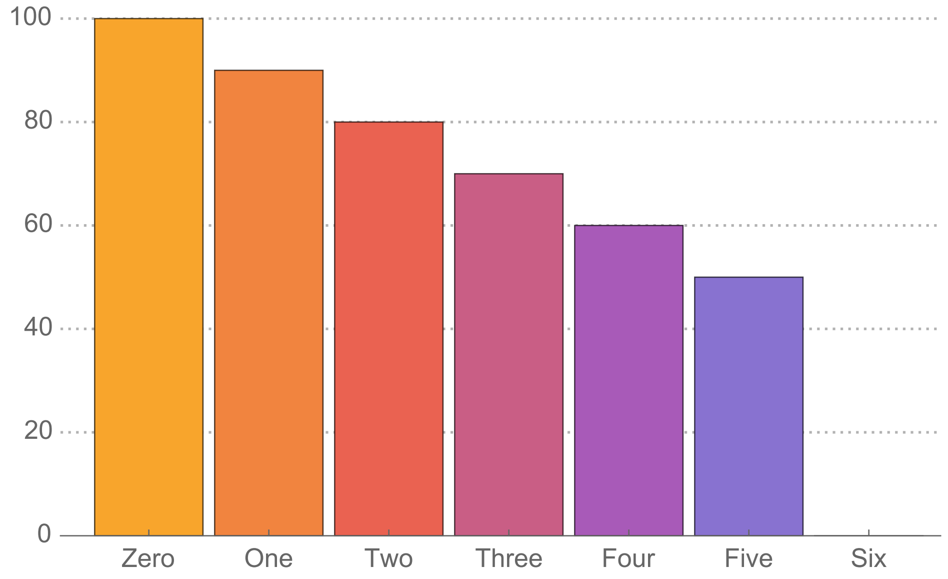
`http://faculty.georgetown.edu/irvinem/theory/Stephenson-CommandLine-1999.pdf.`

You may think—but you would be incorrect—that the command line is out-dated, less powerful, or some other misconception. In reality, the command line is the most powerful since it is a language in which you can express programs about programs. Often, the Graphical User Interface (GUI), is simply a wrapper around commands for the command line. It would be a mistake for you to try to avoid the command line in this course.

We will release the assignments at least a week early, effectively giving you up to an extra week if you complete the previous assignment before it was due. You are strongly encouraged to take advantage of this practice and start your assignments as early as you understand the material necessary to do so.

4.1 Late Policy

The late policy is simple: you lose 10% per day, for the first 5 days, after that you will receive a *zero*. If you turn your assignment in on-time, then you are eligible to receive full points. One day late, you can earn at most 90% of the available points; two days late, you can earn at most 80%. This continues until the sixth day, when you are no longer eligible for any points.



If you have a medical emergency, please provide a note from the attending medical professional. If you have a family emergency, please discuss it with Prof. Long. Depending on the nature of the emergency, documentation may be required. If your computer is lost, stolen, or broken, you can take comfort that since you used `git` correctly you did not lose any of your work.

5 Schedule

Tentative Schedule

Week	Sunday	Monday	Wednesday	Friday
September 22				Introduction
September 29		Getting Started Assignment 0	Using `git`	Programming in **C**
October 6	Assignment 1	Numbers	Control Structures	Loops
October 13	Assignment 2	Functions	Arrays	Numerical Computation
October 20	Assignment 3	Stacks	Recursion	Pointers
October 27	Assignment 4	Bit Vectors	Dynamic Memory	Complexity
November 3	Assignment 5	Sorting	Files	Linked Lists
November 10	Assignment 6	*Holiday*	Hashing	Bloom Filters
November 17		Debugging	Scripting	Trees & Tries
November 24	Assignment 7	Abstraction	Testing & Performance	*Holiday*
December 1		Cryptography	Processes	Review Assignment 8

6 Academic Honesty

The unfortunate reality is that academic dishonesty occurs frequently, and most of the time it happens in introductory courses. Consequently, you will be reading and signing (via `git` submission) a statement regarding this topic. *Your assignments will not be graded until you have done so.*

Academic dishonesty of any kind is forbidden. If you conduct yourself as a person of integrity you will have no problems.

You are expected to adhere to the highest standards of academic integrity. This means that plagiarism[1] in any form is unacceptable. As a (soon to be) computing professional, I encourage you to consult the code of ethics appropriate to your discipline.[2]

All work submitted that bears your name must be exclusively your own. The copying or providing to other students, in whole or in part, solutions (text or source code) to class assignments from any source (including work done by former or current Computer Science & Engineering students, other humans, animals, zombies, sparkling vampires, the Bavarian Illuminati, any of the Four Horsement of the Apocalypse, or materials found on the Internet, *et cetera*) that are not explicitly sanctioned by Prof. Long or the teaching assistants is unacceptable.

All material that did not originate with you must be *cited* – you must credit the source in all cases. You will not get credit for this work, but this is *not considered cheating*. Information and assistance from the professor, teaching assistants, assigned project partners (if applicable), course textbooks, course web site, and code and documentation (*e.g.*, `man` pages) supplied with the **C** Standard Library (`libc`) may be freely used without acknowledgment.

There is no flexibility in this policy. If you cheat, you fail, and your academic career may be prematurely foreshortened. The *reason* that you cheated is entirely immaterial. This policy applies equally to all parties, regardless of whether they are transmitting or receiving the material in question. A more exhaustive description of the consequences and process for academic misconduct appear in the university's policy on academic honesty which is listed at: `https://ue.ucsc.edu/academic-misconduct.html`.

[1] **pla·gia·rize** *vt.* [< L. *plagiarius,* kidnaper] to steal and pass off as one's own (the ideas or words of another) to present as one's own an idea or product derived from an existing source – **pla·gia·riz·er** *n.* (source: Webster's New World Dictionary)

[2] The Association for Computing Machinery is `http://www.acm.org/`, the IEEE is `http://www.ieee.org/` and the IEEE Computer Society is `http://www.computer.org/`.

7 DRC Accommodations

We are committed to creating an academic environment that supports its diverse student body. If you are a student with a disability who requires accommodations to achieve equal access in this course, please submit your Accommodation Authorization Letter from the Disability Resource Center (DRC) to me privately during my office hours or by appointment, preferably within the first two weeks of the quarter. At that time, I would also like us to discuss ways we can ensure your full participation in the course. I encourage all students who may benefit from learning more about DRC services to contact DRC by phone at 831-459-2089 or `drc@ucsc.edu`.

8 Title IX and CARE

We are committed to providing a safe learning environment that is free of all forms of gender discrimination and sexual harassment, which are explicitly prohibited under Title IX. If you have experienced any form of sexual harassment, sexual assault, domestic violence, dating violence, or stalking, know that you are not alone. The Title IX Office, the Campus Advocacy, Resources and Education (CARE) office, and Counseling and Psychological Services (CAPS) are all resources that you can rely on for support.

Please be aware that if you tell me about a situation involving Title IX misconduct, I am required to share this information with the Title IX Coordinator. This reporting responsibility also applies to course TAs and tutors (as well to all UCSC employees who are not designated as “confidential” employees, which is a special designation granted to counselors and CARE advocates). Although I have to make that notification, you will control how your case will be handled, including whether or not you wish to pursue a formal complaint. The goal is to make sure that you are aware of the range of options available to you and that you have access to the resources you need. Confidential resources are available through CARE. Confidentiality means CARE advocates will not share any information with Title IX, the police, parents, or anyone else without explicit permission. CARE advocates are trained to support you in understanding your rights and options, accessing health and counseling services, providing academic and housing accommodations, helping with legal protective orders, and more. You can contact CARE at (831) 502-2273 or `care@ucsc.edu`.