UNIVERSITY OF CALIFORNIA, SANTA CRUZ
BOARD OF STUDIES IN COMPUTER ENGINEERING

CSE013E: COMPUTER SYSTEMS AND C PROGRAMMING

TIME: See Canvas
CLASS: See Canvas
LAB: See Canvas

INTRODUCTION

Computer Systems and C Programming is a class intended to bring you up to speed on programming small and large programs in C. Originally written in 1978, C remains the most popular programming language, and the most used one in terms of numbers of computer programs written in it.

In this class, we are going to approach C from an embedded paradigm, and all of your programming assignments are going to be on a 32-bit embedded micro, the MicroChip PIC32.

You will learn how to program in C, how to write modular code, and some of the tips and tricks when dealing with an embedded micro. This is a programming class and you will be writing lots of code. Expect to spend at least 15-20 hours outside of class playing with the code to get things to work.

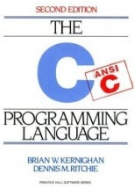
INSTRUCTOR:

MAXWELL JAMES DUNNE
E-mail: mdunne@soe.ucsc.edu
Office: Engineering 2 (E2), 329
Hours: TBD
Phone: (831) 459-4172 (Office)

TAS AND HELPERS:

TA's: See Canvas

TEXTBOOKS:



[K&R]: "The C Programming Language, 2nd Edition" by Kernighan and Ritchie, Prentice-Hall, 1988, ISBN-10: 0131103628. Available in the bookstore.

[Notes]: "Notes to accompany K&R," by Steve Summit available on the class website and at: http://www.eskimo.com/~scs/cclass/krnotes/top.html

Optional Textbooks (good references):

[SummitIntro]: "C Programming Notes: Introductory C Programming Class Notes," by Steve Summit available on the class website and on the web for free at: http://www.eskimo.com/~scs/cclass/notes/top.html

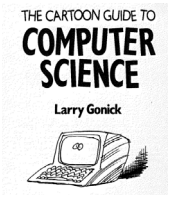
[SummitIntermediate]: "C Programming Notes: Intermediate C Programming Class Notes," by Steve Summit available on the web for free at: http://www.eskimo.com/~scs/cclass/int/top.html

[Wiki]: "C Programming: A comprehensive look at the C programming language and its features," Wikibook, available on the class website and also available online at: http://en.wikibooks.org/wiki/C_Programming

[C-Book]: "The C Book" by Mike Banahan, Declan Brady and Mark Doran, originally published by Addison Wesley in 1991. Available for free online as HTML and PDF versions at: http://publications.gbdirect.co.uk/c_book/

[Hard C]: "Learn C The Hard Way," online set of modules and practice programs that will take you through a fairly complete C knowledge. Available free online at: http://c.learncodethehardway.org/book/ (videos and PDF available but not free).

[Zyante]: "Programming in C," interactive C book online by Frank Vahid and Smita Bakshi (not free, but can be gotten at a discount if you are interested) at: https://zybooks.zyante.com/#/zybook/7VU9pYQ5ue/tableofcontents



"The Cartoon Guide to Computer Science" by Larry Gonick, Barnes and Noble Books, 1983. Out of print, but it is available on the class website in three parts at: http://www.soe.ucsc.edu/classes/cmpe013/Spring11/Gonick/

READINGS

There is quite a bit of material to cover in this class, and you are expected to have read the assigned reading before coming to class. You will get out of this class what you put into it. Simply put, if you do not do the reading, you will not effectively learn the material. We have gone to quite a bit of trouble in order to find appropriate reading for you, so take the time to read them.

The main textbook, *The C Programming Language*, (K&R) is one of the best books you will ever read on a technical subject. Like the language C itself, it is precise and to the point, and assumes you know what you are doing. There is quite a bit of subtle hints on how to code well in the book, and it should be read carefully. The Notes to Accompany K&R point out some of these subtleties and expand on some of the examples. Make sure to read both K&R and the accompanying Notes as assigned (you will most likely want to read them again after lecture).

DEVELOPMENT ENVIRONMENT

We are going to be using a Microchip 32-Bit Processor, the PIC32, for all of the programming assignments in this class. All of our development will be done within the MPLAB-X integrated development environment (IDE), which can be downloaded from the class website (see the Installing Software handout). We are using the Microchip XC-32 compiler for PIC32. We have put together a small kit that includes a PIC32, and IO_Shield with screen, LEDs, and buttons, and a PicKIT 3 programmer so that you will be able to develop on real hardware. We will also be using the built-in simulator within the MPLAB-X IDE.

You will check out the Uno32 kit from BELS (in the basement of JBE), which will charge you a lab fee. When you return your lab kit at the end of the quarter, if it's in working order, you will be refunding most of this. You cannot keep your lab kit and must return it at the end of the quarter (you can buy and assemble your own if you want one).

All of the programming tools are available for free, and can be downloaded from the links on the class website. You are strongly encouraged to install these on your own computers so that you can work outside of the lab sections. Note that while these tools are all supposedly cross-platform, we will only be supporting Windows installations.

The development software is installed on all Windows computers campus-wide, including the 2 24-hour labs on campus. Therefore there is no excuse for not doing your work because your computer died.

Note: we **strongly** encourage you to work in GIT, not just turn in your assignments there. We are not sympathetic to your data loss.

GRADING

This course is based on a combination of the lab and class. They go together and are indivisible. If you cannot complete the lab, you cannot complete the course. This is programming class, and you will be graded largely on the programs you write. Note that we have a strict late policy on the labs, make sure you turn in what you have on time.

COURSE/LAB:	70%	Programming assignments
	30%	In-class quizzes (every week on reading)

In order to pass the class, you will need to demonstrate basic mastery of the subject matter as demonstrated by your labs and quizzes; no one may pass the class with less than 50% average on their lab scores. Note that these are not sufficient to pass the class, but you cannot pass the class without meeting these minimal criteria (necessary but not sufficient).

Re-grading of lab assignments will only be done if we have a clerical error (i.e.: we added points wrong) or we somehow missed your work. Note that this does not include you submitting the wrong commit ID on CANVAS. We will allow a single exception to resubmitting the correct commit ID per student once per quarter. If you feel your lab has been graded incorrectly you must request a re-grade within one week of the release of the grade.

Note: there is a sure fire quick and easy way to fail this class—that is to wait until the day before the programming assignment is due to start it. There will be a new programming assignment every week, start it as soon as you get it.

LATE POLICY

Late labs will be penalized 10% per day (counting a partial day as a whole) they are late. After 5 days, giving a maximum score of 50%, you will receive a 0. This late penalty is processed by CANVAS, give yourself some leeway when submitting assignments to account for internet issues. If there are extenuating circumstances please contact Max Dunne ASAP, it is likely we can work something out but this only really works if we are informed in a timely manner.

GETTING HELP

This is class covers a broad swath of topics; there are resources available to you for help. I will hold office hours every week. The TA will be available in lab sections and if necessary hold their own office hours.

Lastly, there is a Piazza forum for the class that is quite active and monitored by the TA and the instructors. If you have a need to contact the teaching staff, do so with a private post on Piazza. Any email request will be bounced to Piazza, and answered there. Likewise if a question has already been answered on Piazza, your question will be referred to the already posted answer or simply deleted. Learn to use the search functionality before posting.

Don't post code onto Piazza; if you need one of the teaching staff to look at your code, post it in a private message to the teaching staff.

If you do need to email the class instructor, use the mdunne@soe.ucsc.edu address. Emails will be checked at least once per business day; expect an answer by the end of the next business day. Piazza is a much faster form of communication. If the request is one that should have been handled on Piazza it is likely I will not respond via email other than to tell you to post on Piazza.

Note that Piazza is a public, professional forum; please treat your communication with others with a modicum of dignity and respect. You are not anonymous. You ARE training to be a professional, treat each other with respect.

UC Santa Cruz is 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 this 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 by email at drc@ucsc.edu.

LECTURE VIDEOS AND SLIDE COPIES

The class lecture will be recorded and the slide decks recorded using the screen capture capability of the Surface computer. These will prove to be useful when reviewing the material for midterm and final, or merely to go back and go over material to refresh it in your mind. However, if attendance drops, the videos and class slides will be uploaded with a delay (likely a week).

COURSE WORK:

Attendance is highly recommended for the lectures as the material builds up quickly. The class lecture will be recorded and the slide decks recorded using the screen capture capability of the Surface computer. These will prove to be useful when reviewing the material for midterm and final, or merely to go back and go over material to refresh it in your mind.

There will be weekly quizzes in CANVAS that are both required and graded. They are worth 30% of your overall grade, and they are essential to mastering the material. In my experience, understanding the material requires at least three passes—(1) reading the material before lecture, (2) attending lecture and paying attention, and (3) synthesizing the material into your own work during the programming assignments. Skipping any of the steps requires a whole lot more work to make it up on your own.

LAB WORK:

This class should be a lot of fun. I would add also a lot of work (think: drinking from the fire hose). There is a lot to cover, and only 8 weeks to get you familiar and confident in programming in C. Prepare to spend over 20 hours a week on this class, less at first, much more towards the end. We will make every attempt to help you, and to ensure that you succeed, but you have to put in the work yourselves.

In addition to the class lectures, you are expected to attend lab section but they are technically voluntary. **If the sections are overcrowded, priority goes to the students signed up for that section;** you are free to go to any additional lab sections as space allows.

We will be working with a 32 bit microcontroller, the MicroChip PIC32, using a development kit manufactured by Digilent (students who wish to buy their own dev kits can purchase them directly from Microchip or Amazon). These kits will be repurchased from you at the end of the quarter if they are still in working order. See the section on Development Environment above for details about the tool chain. It is strongly recommended that you install the full development environment on your own computer. You will be working on it quite a bit, and it is nice to be able to work anywhere.

As a programming class, you are going to be doing a lot of programming. The bulk of your grade in based on the programming labs. As such, you will be helping each other out in the labs to understand the material. **However, this is NOT license to copy others' work**. Credit for collaboration should be explicitly noted; failure to give credit on collaboration is considered a

form of cheating and will be dealt with accordingly. **Writing code together is strictly forbidden unless explicitly allowed**.

Lab assignments are to be submitted via GIT. There is no other acceptable way to submit an assignment unless specified by an instructor. Details on how to do this are CANVAS.

LAB SECTIONS:

Look on AIS

There are two lab sections per week starting on Thursday. You are required to enroll in one, and *we will make sure there is space for you in that section*. During the lab sections, the tutor and TA will go over tips and tricks, what is required of you, and help you code your assignments. You are welcome to any of the other lab sections, but priority is given to those enrolled in that specific lab section.

Note that lab section is when you have access to the TA and tutors, most programming assignments will require you to work on your own outside of those times in order to finish the programming assignment. Git should make this easy to do this as long as you remember to push before finishing working.

Assignments are designed to take more time than your lab section to complete; the lab section is to give you guidance and help, it is expected that you will go off and program on your own both before and after sections. The total time given to complete the labs is sufficient if you start early; *starting just before the lab is due is a recipe for failure*.

You are expected to have read (not skimmed) the lab manual before lab section. During lab sections the teaching staff will have to manage the load of more students asking questions than there is time or staff to handle. This is especially true of lab sections that immediately precede the due dates of the assignments. If the question you are asking is covered in the lab manual, or the nature of the question makes it clear that you have not read or merely skimmed the lab manual, the staff will turn you away and tell you to read the manual. In the vernacular, this is often called “RTFM”, for “Read The Frakking Manual.” Note that if you have actually read the manual (not skimmed) and need clarification on something that is not clear, the staff will be happy to help you.

Lab extensions will rarely be given, and will only be given if we have sufficient evidence from the GIT commits to demonstrate that you have (collectively) been working on it the entire time. Learn to commit early and often. If we see evidence that too few started the lab early (in the form of commits and pushes), then no extension will be considered.

You will be submitting your work using Git. Be sure you understand how Git works BEFORE the deadline for the first lab.

USING GIT

We have a supplementary document on using GIT, which explains the mechanics of using it specifically in this class. GIT is an industry standard version control tool, and will prevent you from having data loss in your coding assignments. It is also how we will be having you turn in your code for grading. Pushing the code to the server safeguards it from loss, and the Commit ID tells us which version of your code you would like us to grade. Note that both are required to turn in your assignment.

You are encouraged to commit and push your repo early and often. There is no penalty for doing so, and it will make sure that you have met the minimum file requirements for the lab. Furthermore, we will be monitoring the activity on the server when deciding if a lab extension is merited. If you don't push your code, we have no idea you have been working on it. Learn to use GIT wisely, check your code in frequently (and start it early). A small portion of your lab grade will be based on how often you committed your code using GIT.

ACADEMIC HONESTY

Academic honesty is a requirement for the course. All assignments must be your own independent work; this includes both quizzes and programming assignments.

What is cheating? It is presenting work that is not yours as your own. You can—and are encouraged to—discuss and strategize with your colleagues on the material and labs, but **your work should be your own**. Copying is **NEVER** acceptable.

On the labs, **cheating is sharing code** unless explicitly told that it is permitted. If a student is caught cheating in either the class or the lab this will result in an immediate failure in the class and the lab. It will be reported to your college and your department. **DO NOT CHEAT**; it is not worth it.

ACKNOWLEDGEMENTS

I would like to acknowledge Cyrus Bazeghi, for all his help with the course material, organization, and lecture material. Gabriel Elkaim, who renovated this class for the PIC32 and allowed me the opportunity to teach it. Steve Summit, who taught an Introductory and Intermediate Programming class at the Experimental College at the University of Washington in Seattle, WA, has generously allowed us to use his notes on K&R and other supplementary materials. Microchip Corp. has generously provided slides and software through their academic partner program, and some of the slides on file systems come from Henry Cheng at UC Davis.