

Answers to your questions

(minus those I answered in class or which are already answered on the syllabus—I also combined some questions that were essentially the same).

What interests you most about low-level programming/what got you interested in it to begin with?

I like all programming, but low-level programming allows us to interact with the hardware in interesting ways — especially when I am also creating the hardware. It might say something about my personality, but I like having full control of everything, and seeing how everything actually works under the hood.

What is your computing spirit animal and why?

Ah, touche! The hunter is now the hunted.

I'm not 100% sure, but here are some candidates:

- **Turtle** — I like to dictate the pace of my existence, and I value slowing down and thoughtful persistence. I prefer deep fundamental understanding over chasing shiny things. I would rather take my time and build something great rather than hastily throw down a dirty hack — though, you know I've created my fair share of dirty hacks.
- **Dolphin** — I'm curious and like to explore and play, and I'm a fairly social creature.
- **Dragonfly** — I do like to kill bugs.
- **Salamander** — I like to hang out at the software hardware boundary like an amphibian enjoys both water and land.

Do you still have professional goals outside of teaching?

I have technological goals. I want to fundamentally change the Internet of Things to make it more usable, more sustainable, and just basically something we won't regret 500 years from now.

I don't really aspire to any specific level of fame or leadership position — administration doesn't sound particularly exciting to me, though I would do it if I needed to to benefit my organization. I also don't have specific money/income goals. Money is great and useful, but I tend to focus on what will allow me to do the work and play that I want to do, rather than what will make me the most money.

Have you ever considered putting more time/effort into content creation? (Making TikTok/YouTube Shorts videos, big projects, etc)?

Sure. I think about a lot of things, but can only juggle so much while teaching classes and running a research lab.

Your favorite memory related to the field you work in?

It's hard to pick just one. Here are two candidates.

- During my Ph.D. I worked with some biologists studying animal behaviors. We worked with tortoises, sea otters, and mongooses. It's more of a collection of memories, but I loved being able to take my computer science out into the field and pair it with my love of the natural world.
- In early 2020, I had the opportunity to help organize a hackathon in Gaborone, Botswana. During the hackathon, I taught an electronics and microcontrollers workshop and there was a team of students from a local high school — who had little programming experience beyond hello world — that just light up. It was like they had discovered sunlight for the first time as they realized that they could tinker and create new technologies. That was pretty special.

Where do you want to end up before you retire? (If you retire).

I'm really not sure if or when I will retire. I think the thought of me without a day-job to keep my hobbies in check terrifies my wife. But, seriously, I might retire from being a professor, but I doubt I will ever really retire if I can help it.

I've so far been able to help found/create a new research community (the batteryless computing community). We're still in our small/crazy idea phase, but growing. I would love to see batteryless computing become a mainstream commercial reality before I retire.

Favorite place to ski?

Utah in general. Downhill, probably Alta or Snowbird. Nordic skiing—any out-of-the-way trail. The Uintah mountains are full of great candidates.

Is there any specific reason why you chose February 17th as the day we have to go to office hours before we lose all our participation points?

It's pretty arbitrary. I like to set it early enough in the semester that it's helpful, but not so early that we end up with unnecessarily crowded chaos.

What do you enjoy about coding? Were you always good at coding?

I like that it combines aspects of problem solving and creativity. It also lets me get involved in interesting things.

No, like everybody else, I was disastrously terrible at programming when I started. But, I didn't know how bad I was, because I was learning on my own (I was better than my family members). I also really wanted to be good at it, and kept working at it until things got easier.

Thoughts on dividend stocks?

Umm....I like them, if the company is solid.

I am curious how your research led into embedded/low-power system development, or, if it is something you started in, why you decided to work in that subfield to begin with.

I was initially interested in distributed systems and distributed problem solving. My first project in graduate school involved rethinking how we design laptops for better energy efficiency. Interestingly, we didn't really change how laptops are designed, but a variant of the technique we published is now used in nearly every smart watch (and many mobile phones) on the market. That project got me hooked on energy and the challenges that we face when dealing with energy scarcity. That led me into energy harvesting and embedded applications where human intervention isn't always possible. And, here I am.

Why software rather than hardware/electrical?

It's just what I discovered and fell in love with first. I didn't really discover hardware and electronics until grad school.

When you were in school did you have any projects you worked on outside of class that you enjoyed?

Yeah. In my last year of undergrad, I did a research project where we were developing distributed algorithms for finding bugs in communication protocols. The algorithms were inspired by honey-bee foraging behavior, and managed to find errors in seconds that had previously taken months to find. That's what got me thinking about graduate school.

If you could provide any piece of advice (life or school related) to us students what would it be?

The important is more important than the fun, but fun is also important.

Do meaningful things with your life. Develop rare and valuable skills. Work hard and smart, but don't forget to enjoy the journey. Also, make sure it's your journey, and take care of the people around you.

What is your favorite thing about Clemson?

There are a lot of things I like about Clemson. The people (well, most of them). I've been able to work with some exceptional students. I like that the faculty have a culture that is more cooperative than competitive. It's been a great place to raise my kids. I like the weather and the fact that I can garden all year long. Great blue cheese, as well.

Do you have any interest in a particular form of art?

I like art. I went through a weaving phase when I was a teen — built a loom, wove some small tapestries. Also, a watercolor phase. I like photography and filmmaking. I like theater (both watching and acting).

Have you always been interested in batteries and energy or specialized once you worked for awhile?

Since graduate school.

What's your favorite microcontroller?

Probably currently the MSP430FR5994 (Texas Instruments). At least it's the one I use the most, these days. I'm really excited about some of the new Ambiq MCUs.

What's your favorite movie?

I like movies. I'm fairly easily entertained. It's hard to pick just one. "Waking Ned Divine" is pretty great, though, if you're looking for one that most of you probably haven't seen.

Do you like Clemson?

Yes, I do.

What is your IQ?

No idea. I've never had any interest in trying to describe mental abilities with a single number. It seems pointless except to try to rank people, which seems...well, pointless.

Have you ever done any software development work outside of Clemson? Like contract work or any really cool projects?

I did some contract work before coming to Clemson. While at Clemson all of my software development work has either been my research projects (I think some of them are pretty cool), stuff for my classes (like my autograder), and a few hobby projects.

How many hours would you recommend be dedicated to studying for quizzes/exams? I get over half of the grade is projects but will there be a need for intensive studying or is most of the time more likely dedicated to the assignments?

Time needed is going to vary from person to person, but I recommend studying incrementally, as we go. Study new concepts right after we discuss them. Try to come up with example questions that I might ask about the new concepts. Ask questions if anything isn't clear.

Studying before exams could be helpful, too, but I find that people who study incrementally don't need as much cramming and exam prep at the end. And, they tend to do better on the exams.

Favorite programming language?

That's a tough one. It depends on the job at hand. For low-level, or embedded stuff, probably C. For high level OO-style programming, probably Ruby. For quick and dirty shell scripting, either ruby or python.

How many languages do you know?

Depends on what you mean by "know"? I've done interesting nontrivial things with the following: C/C++, Ruby, Python, Javascript, Java, Pascal, Basic, R, Assembly (a few different flavors), C#, PHP, Haskell, Scheme, Bash (and other shell languages). I have played around a bit with others, like Rust, Kotlin, Go, OCaml, Objective-C, Lisp, Perl, Racket, and Smalltalk. I'm sure I'm forgetting stuff. I've also created a few languages of my own (Eon, Mayfly) both for research purposes — not really recommended for public use.

Most hated programming language?

I have a hard time hating any of them. I mean, I don't want to program in Perl if I can help it, but it has a certain charm that I can't help but appreciate.

Your SSN and credit card number?

555-55-5555 & 1234-5678-4444-0000 (enjoy).

What was your favorite class in undergrad? Favorite class in grad school?

Three-way tie. Operating Systems (seriously), Algorithms, and User Interface design. My networks class was pretty great too.

Grad school is a bit different. Classes aren't so much the focus, but David Jensen's research methods class was a definite highlight. Micah Adler's Algorithms class made me question my sanity and value to the world as a computer scientist.

What classes have you taught at Clemson aside from this one?

Embedded Systems Prototyping, Graduate Embedded Systems. Software Dev. Fundamentals (CPSC 2150), Networks/Network Programming (CPSC 3600). I also lead the Embedded Networked Systems Seminar and the School of Computing's New Ph.D. Student Seminar.

What is your favorite course that you've taught?

It's tough to pick. I love teaching this class. It's probably tied with Embedded Systems Prototyping, which I teach less often.

What's something you've never taught that you really want to?

I would love to teach Computer Organization/Architecture (CPSC 2310), but an embedded systems-oriented version.

What made you want to start a YouTube channel?

Like I mentioned in class, it started with 3220. Well, I first created a channel so I could share little videos my kids and I made (privately) with family members. Then, I was traveling for a conference, and recorded a video for my 3220 students to help with one of their projects. They liked it, and so I decided to make more, and things just sort of evolved organically from there. It's been fun. It helps me learn new things, and connect with a lot more people.

What was your favorite past-time in college?

I did a few different things: indoor soccer, fly fishing, hiking. In grad school, I picked up cycling and cheesemaking.

What is something you do to avoid tech burnout over time? You seem to enjoy Comp Sci and I would like to maintain that enjoyment also.

Hmmm...choose to work for organizations with great cultures — organizations that look out for their people. Also, developing strong tech skills helps, because you have more choice in what you do for work, who you work for, and you often can do more in less time, all of which help avoid burnout.

How do you define success in your course?

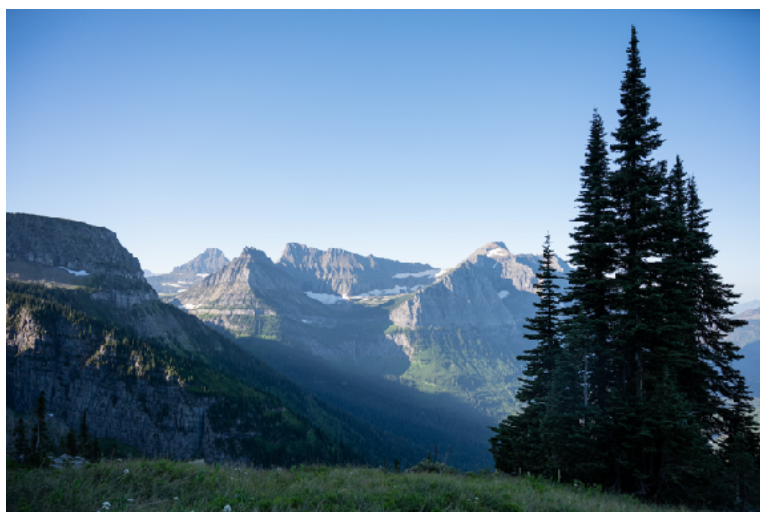
Well, I assign grades based on the syllabus, but I consider myself a success when—after the experience—you tell me that your life/job/confidence/prospects are better because of this class.

How do you best build relationships with students?

With 50–100 students, this is tough. I require office hour visits, in the hope of talking at least once with all of you. I try to take time to talk with students about things both in and outside of class. I try to give you work that eventually blesses your life (in the moment, it might not always feel like a blessing).

Do you like to travel? Where have you been? Where is your favorite place you have traveled?

Yes. A lot of places. That's almost impossibly tough. Highlights outside the US include...Cambodia, Botswana, Zimbabwe, Namibia (photo), Korea, Belize, Sweden, Germany, France. Glacier National Park is one of my favorite places inside the US (photo below).



How difficult do you usually try to make your projects? Some classes the projects take a couple hours, others are 50+ hour chores, and some classes do both throughout the semester, but the professor usually has expectation for how long the various projects should take to complete.

The time the projects take depend on your preparation. I will give you roughly 3 weeks per project, and I expect them to take the average 3220 student one week (maybe 10–20 hours) to complete. That said, I've seen students finish projects in 3-4 hours, and I've heard people say they spent 50+. I would expect the middle two (#2 and #3) to be the most challenging. That said, people often don't take me seriously on project 1, and so project 1 usually ends up doing the most damage to grades.

What (in your opinion) has been the coolest thing a student of yours has gotten involved in?

That's tough. A lot of my students work on stuff they can't tell me about. My students have worked on software for airplanes. One is designing technologies for microsatellites. One of my students (Josiah Hester) is a professor at Georgia Tech, working on a bunch of really cool batteryless tech for healthcare.

A few of my former students created the Clemson Makerspace and CUHackit, which are both really cool in terms of their potential to transform Clemson for the better.

What is the most interesting topic in this course? What is the hardest topic you will cover, this year?

I have a hard time picking a favorite. Threading and concurrency, memory management, and file systems are all great (I know I just described 75% of the course, but that's the best I can do at choosing a favorite).

As for difficulty, there is a lot of variance here. Threading and concurrency usually causes the most difficulty (but some have more trouble with scheduling and file systems. Memory management (paging, segmentation, and address translation) can sometimes take students a little while to wrap their heads around, but it's not really hard, once you do.

What will the programming assignments be like?

They will be small programs or software libraries, probably a bit more complicated than those you saw in your 2000-level courses. I will give you a functional description, the autograder, and three weeks to complete the project. Each project will be designed to help you better understand some facet of the class (like scheduling, resource management, concurrency, mutual exclusion, or file system structure).

Respect the assignments and start early, and you should be fine.

How would you suggest that we prepare for quizzes? Are you mainly pulling content from in-class discussions, or the textbook reading?

I pull content from both the discussions and the readings, with emphasis on stuff that we have both discussed and read. Prepare for quizzes by studying right after class meetings and using active recall (as discussed in class on the first day).

Why did you make the autograder? How long did it take you?

Students were always super stressed about projects, because they had to wait and didn't know if their code crashed and resulted in a bad grade. Also, manually grading projects is mind numbing. I feel bad asking a TA to do that. So, it just made sense. Version 1 took me a week during the holiday break back in ... 2014 (I think), and I've done a lot of work on it since. Totally worth it, though.

Did you learn most of your coding from classes or on your own?

I learned a lot from my classes. And, I've learned a lot on my own. I don't think you really know much in CS until you do it (or use it). Many of my classes gave me a lot of hands-on experience, and helped me see what was possible. My various jobs and projects have taught me a lot as well.

Are you a sports fan? Favorite sports?

I like sports. I'm definitely not a super-fan, but you will occasionally find me watching or playing something.

Favorites:

To play—soccer, running, cycling, ultimate frisbee.

To watch—soccer, college football (when the teams are good), high school track and cross country (when my kids are competing).

What is your favorite thing about programming?

The combination of a puzzle (the challenge) and creativity. I like producing things that make things work better (and feeling like I can speak an alien tongue).

What is the coolest thing you have made?

My three daughters (can only take partial credit, though).

My first successful wheel of mountain-style cheese (Emmenthaler) was pretty cool. Mostly it was just really cool that it actually worked.

I'm also fond of the chicken coop that I built with the automatic solar-powered door (to keep the raccoons and foxes out). I'm currently building an aquaponic greenhouse.

Most challenging project?

Being a father. It's also the most rewarding project, so far.

Do you have dogs? Why or why not? Pets?

No dogs. I like dogs, but I prefer them to belong to someone else.

For a long time, my basic pet philosophy was that I prefer pets that 1) do something useful for me or my family, and 2) have a straightforward (and legal) way out for me, if my kids stop helping me care for the pet they asked for.

Naturally, I eventually broke down, and was adopted by an abandoned cat (The Great Catsby). I'm definitely his favorite family member (with zero effort on my part), which really bugs my 15-year-old daughter, who won't leave the poor guy alone.

I also keep chickens (see #1 and #2 above), bluegills, and shubunkin goldfish (still trying to figure out how to get them to do something useful).

I think it would be great if someone would develop a breed of pigmy cattle or bison that you could raise in a normal-sized backyard. They would mow your grass and provide meat and milk, and you could claim it was an odd breed of dog if the neighbors started asking questions. That might be my ideal pet, if it existed and wasn't an inbred fragile genetic mess.

Why teach OS (assuming you have a choice)? Why do you love OSes so much?

Yes, I do have some choice in the matter. I love the topic. It's close to my research. I like low-level hands-on classes, and OS is both low-level and hands-on. The OS is the foundation on which we compute (if you don't count the hardware). Improvements at that level have impact far beyond a single application.

Everybody is acting like you're at least minorly famous? What's up with that?

I really don't know. People definitely don't recognize me when I go places.

Why do you think you have a reputation as a “hard” teacher?

It's one of life's great mysteries. I think students learn the most when challenged. And, so I try to challenge you. Past students tell me that they worked hard in my class, but felt that they came out better prepared for life and career.

I have been told that I write challenging tests, though students seem to do fine on them (no shortage of As). One thing that you will find, is that I try to write test questions that evaluate your ability to apply the principles and tools from the class. I don't like pure memorization questions, because out in the real world, memorization doesn't get you very far.

I have also been called “hard” because I don't accept late work and will give you a zero (0) for code that almost works, but doesn't compile because you forgot a semicolon. Start early and test your code, and I think you will find me to be a fair and reasonable grader.

How did you get into computers? Why CS?

Very cliché. In the 5th grade I wandered into the computer lab one afternoon and noticed some of the older students (11th or 12th graders) programming. I asked them about it. One of them gave me a Pascal compiler (on a 3.5-inch floppy disk). I took it home and played with it. I eventually moved on to C/C++, and the rest is history. I've always enjoyed how computing is both creative and systematic. I also love that I can use my skills to work on just about any of the world's many problems.

What did you struggle with most when you were a novice computer scientist like us?

I struggled with the same stuff you all do—pointers. :-) Well, that and understanding scoping and stack/heap issues. Back then, the main difference was that it was much harder to find good documentation. The Internet was barely a thing. So, finding the right book at the library was a challenge, as well.

What you do with programming outside of teaching computer science at Clemson?

Well, I do a lot of research (see other answers). I also have some hobby projects that I work on. In recent years, I've helped with a programming and electronics course at RC Edwards Middle School. I try to teach my kids a little programming, as well, when they're interested.

What research do you do? What are your research interests? What is your specialty?

My research focuses on mobile computing, sensor networks, and embedded systems. Specifically, I try to make small, flakey, resource-constrained computing devices perform better and do more, in spite of their shortcomings. My current projects are many. I have developed wildlife tracking technologies with the USGS (desert tortoises and sea otters). I am working on security and privacy issues in wearable medical devices. I have worked on making manufacturing environments smarter and more efficient. I'm also generally interested in exploring techniques for making RFID-scale batteryless devices more generally useful, even though they reboot all of the time and have trouble keeping time and making forward progress. If you are interested in research, and this sounds interesting, then we should talk.

Do you do research with undergrads?

Yes, and quite a few former 3220 students have worked in my lab. I usually have between 1 and 5 undergraduate students working with me, I currently have 3. Some of my former students are working at places like Apple, Google, and Amazon, as well as some smaller startups. Others have gone on to Ph.D. programs at places like CMU, Dartmouth College, and the University of Michigan.

Why did you choose to work in academia, rather than industry? I assume industry would be more lucrative.

Interesting question.

When I started graduate school, I was sure that I was headed to industry after my PhD.

But, I found that my research fits best in academia. I also enjoy the intellectual freedom, and I really like working with students (both undergrads and grads). My experiences in industry were all positive. The money was fine in both. But, you really don't get a PhD for the money.

I think the following article (by Nick Feamster at Univ. of Chicago) spells it out pretty well.

<http://greatresearch.org/2013/08/30/industry-or-academia-a-counterpoint/>

What kind of music do you enjoy? What are your musical interests?

A wide variety. It's always changing, but today I was listening to Grits, Matt Kearney, Dr. Dre and Snoop Dogg, Counting Crows, The Mighty Mighty Boss Tones, Pentatonix, Matisyahu, Assorted Khmer music (wishing Spotify/Apple Music would add some Meng Keo Pichenda), John Rutter and the Cambridge Singers (classical choral), and some gregorian chant. Last few years, my kids brought home a lot of Imagine Dragons, Taylor Swift, and the Hamilton soundtrack. These days, they're in a classic rock phase that's proving interesting. My daughters also recently discovered that I was into ska back in the 90s, which has led to some interesting walks down memory lane. I like music that inspires and

energizes me. I also like music that celebrates the things that are great and noble about the human experience. I'm also an OK singer, a rusty pianist, and occasionally a novice composer/arranger.

Favorite type of food?

Hard to say. Khmer, Thai, Korean, Indian, Persian, Greek, Mexican and French are all outstanding. But really, I just like good interesting food.

What's your favorite fast food place, and what's your order?

Hmmm. I'm not really much of a fast food fan. Maybe the taco place in Central (Taqueria la Texana). I would order an assortment of tacos (probably Asada, Pastor, and Chorizo) and maybe a Tamale.

Any interesting opinions on A.I.?

My opinion is that AI is interesting.

Where were you born? Where did you grow up? Where have you lived? What was your favorite place to live?

Born in Utah. Grew up mostly in Utah, with a few brief stints in NV and OK when I was little (my Dad was in the military). I've lived in Utah, California, Nevada, Oklahoma, Massachusetts, New Hampshire, and South Carolina. I also lived in Cambodia for two years, Germany for a summer, and spent the 2019-20 academic year living in Botswana. I've thoroughly enjoyed all of them that I can remember (all but NV and OK). Can't really pick a favorite, but Cambodia was probably the most life-changing.

What did you want to be when you were young?

How young? Early childhood, I was interested in being a scientist, astronaut, or forest ranger—maybe occasionally a professional soccer player. I've wanted to be a technologist for a long time—it started coming on in high school.

What is your least favorite student behavior? Pet Peeve?

Cheating is the worst. That would probably be followed by students trying to avoid getting an education and complaining.

What industry jobs did you have before Clemson?

During college, I worked for IBM (first as a tester and then as a developer), and then for two startups: one that provided an online document management service for companies, and another that made road-side radar sensors for traffic counting. At the latter, I wrote a little firmware for the sensors, but

mostly developed traffic prediction algorithms and code that handled the data from the sensors. I also got to build a video sequencing server that put news shows together for a local TV station (but that's a long story). After graduating, I worked for another startup, that made diagnostic software for automotive technicians. During my PhD, I also worked for one of Intel's research labs for a summer.

How do you stay captivated by operating systems and "lower-level" concepts when new technology is emerging? Personally, I don't like C, assembly, and computer organization because these concepts seem obsolete in modern-day Software Engineering jobs.

It's the foundation for all of those new emerging technologies. So many high-level software decisions are affected by what happens in the layers below. Understanding how they work will make you a better developer, even if you never use them directly in your job.

What has your career path been like up to this point? What experiences can you share?

I'm sure you'll hear about some experiences throughout the semester. It's hard to even know where to start. I mentioned my industrial jobs in a previous answer. When I got out into the workforce, I worked with a few PhDs (machine learning experts). It didn't take long for me to notice that those guys were having more fun. They were working on more interesting problems. My work life wasn't bad, but I started to think I wanted to be doing more bleeding-edge type stuff. So, I decided to go for a PhD. After my PhD, I worked as a postdoc at Dartmouth College for 2-years while the recession ran its course, and then got a faculty job at Clemson. I was also a visiting faculty member for 8 months (right before the pandemic) in Botswana.

What does research into embedded systems actually entail? What aspects are under research? I don't know too much about it.

A full answer to this one would be LONG. My lab mostly works on ultra-low power designs and batteryless designs and all of the challenges that come along with having a very few resources and an unpredictable energy supply. Happy to talk more about it if you're interested.

Do you speak any foreign languages?

German (a little and very rusty), Cambodian (reasonably fluent with everyday conversation topics), a very little Setswana, and a phrase or two in Spanish and Vietnamese.

What school did you go to?

Undergrad: Brigham Young University (BYU)

MS & PhD: Univ. of Massachusetts Amherst

Post Doc: Dartmouth College

What was your major? Is your background CS or EE?

Computer Science. I consider myself a self-taught electrical engineer, as well, though I don't have a formal EE education. I've just done a lot of hardware hacking in my research, because I couldn't buy the hardware I needed. So, I learned how to make it.

Whats your favorite breakfast cereal? What do you eat for breakfast?

I actually make my own. I know it's maybe a bit odd, but I buy grains, seeds, and nuts in bulk and mix my own whole-grain cereal. I also make homemade granola occasionally. I love eggs (from my chickens). I make sourdough bread (usually weekly) and sometimes have that for breakfast. I also try to include greens from my garden whenever I have them.

Hobbies?

[In no particular order] gardening, running, cheese making (and eating), cycling, hiking, cooking, playing soccer, travel, playing the piano (badly), singing, composing/arranging music, flyfishing...oh and yes programming and electronics. I used to be a bee keeper. I keep chickens — my current batch is the loudest and stupidest I've had yet.

If you never had to work again what would you do?

I might still do computing research and teach students—I really do love my job. I might negotiate away some of the less fun parts. I might also take any one of my hobbies to an unhealthy extreme. I think the thought of me retiring makes my family a bit nervous.

What is the worst piece of advice you have ever been given?

It's hard to know what the **worst** one was.

I was told that research on batteryless intermittent computing was too hard and that I should switch to something less risky. Now, it's the bulk of what I do. I'm really glad I didn't follow that advice.

Did you always want to teach? How did you get into teaching CS?

Nope. I was convinced that my career was going to be in industrial research, until half way through graduate school. I just didn't have enough experience until then to make an informed decision. I just realized that 1) I love to teach, and 2) that my research fits better in academia than industry.

Do you enjoy teaching this course?

I do. It's one of my favorite topics.

What other field would you be in, if you weren't in computer science?

Do computer engineering and electrical engineering count? Probably something medical, math, or physics, but I'm really glad I'm in CS.

What advice do you have for someone interested in a career in academia?

Start planning now. Get advice now, from several people. Start doing research, now.

Seriously, there isn't enough room in this document for all of the advice you need to hear, but too many students wait to start asking for advice until it's too late to do much about it. Let me know if you want to chat about it sometime.

What is your passion?

I have a few. My family. My faith. Science and Teaching. Food. Growing things. I really think I mostly want to do things that make the world better for others.

What are your spiritual background/religious beliefs?

That's a tough one to summarize quickly. I'm a member of the Church of Jesus Christ of Latter-day Saints. That said, my faith journey is a bit complicated. I don't mind talking about it, but I don't think I'll go into details about it here.

Other than OS/Embedded Systems, what is your favorite CS topic?

That's a tough one, akin to asking me to pick a favorite parent. I would probably have to say it's a toss up between programming languages, compilers, and algorithms. I think machine learning is pretty cool, too.

How did you end up at Clemson?

I actually originally didn't realize that Clemson had Ph.D.-level research in computing. I didn't even know what state it was in. Then I met a few Clemson faculty members who were doing cool stuff. The school of computing posted a job. I applied, and they gave me a job. In the end, I chose Clemson because of the people. There were some key faculty members that I could collaborate with, and both faculty and students seemed genuinely happy to be here.

How long have you been teaching? How long have you been a professor?

Since 2012 at Clemson. Two years before that at Dartmouth College. A little bit before while in grad school at UMass Amherst.

How much do you enjoy making YouTube videos? You said you started it to help teach your Operating System students, but it seems like it sparked an interest in you.

Yeah, I do like it. I've been interested in photography for a while, and I like playing around with the processing, lighting, and audio. Seeing how things play out with the YT algorithm is interesting. And, I also like that it helps me reach a wider audience. It's a pretty solid mechanism for recruiting graduate students.

What is it like to have a YouTube channel and what do you want to see improve about it?

It's like having a puppy that's usually fun and cute, but occasionally bites you and always needs to be fed. :)

There's so much I could improve. I don't know where to start. I would love for it to grow enough that I could pay someone to do the video editing.

YouTube gold play button when?

No clue. At the current rate, probably sometime around 2043.

If your YouTube channel became successful enough, would you consider quitting teaching and doing that full time?

Definitely not in the short term. I like research and being in the classroom too much. But, I don't like to say that I would never do something. So, maybe...

Do you post lecture slides online?

I don't post slides. I don't really use slides. I used to a long time ago, and I found that when I post slides, students disengage (assuming they can just get the details later). Unfortunately, when they try to get the details later, there's nobody to clarify, and they don't learn it as well. That said, I don't mind if you post your notes on Piazza.

Is there a way to get involved in your research?

Yes, maybe. Let me know that you're interested. Show me that you're a strong programmer? Learning a little about hardware can be helpful (not always necessary)? Then if we can find a project that we're both excited about...sure. Very doable.

I'd love to know about the auto-grader. It seems pretty intimidating if I am honest.

What do you want to know about it? It's just a collection of python scripts (and some C programs) that grab your latest submission, compile your code, and run a set of tests on it, and then email you the results.

Please don't be intimidated. It's there to help you. As long as you start early so you can use it a few times, we can discuss anything that seems confusing about it.

Does your autograder test for malicious software?

No, but it keeps good records and has some limited protections. I have never had a problem with people trying to hack the autograder. I think most students would have trouble pulling it off without getting caught. Please don't try it. You'll make more work for me, and it will probably not end well for you.

Have you worked in any other fields?

One of the things I love about computing, is that it makes it easy to work in just about any field. In the course of my research, I have worked with medical doctors, biologists, automotive technicians, transportation engineers, civil engineers, and farmers. Does that count? I also worked in landscaping and as the freezer manager for a local grocery store in a former life.