Teaching statement
Trinabh Gupta, trinabh@cs.utexas.edu

Among the many benefits of an academic position, the opportunity to teach and guide students stands out for me, for various reasons. First, I enjoy the responsibility that comes with teaching—the sense that you are the one to educate, motivate, and direct students. Second, the success of mentored students can impact society, which is satisfying. And, third, teaching helps push research forward. For instance, one of my PhD advisors regularly recruits students (both at the undergraduate and the graduate levels) from his classroom and involves them in research, with successful outcomes.

My teaching style and philosophy are heavily influenced by my own experiences as a student, mentee, teaching assistant, mentor, and presenter (at conferences). Overall, my philosophy emphasizes conceptual clarity and accommodating students’ diverse backgrounds.

Experience
Teaching. In fall 2011, I was a teaching assistant for the Introduction to Operating Systems course (with just over a hundred students) taught by Prof. Aloysius Mok at UT Austin. The duties included answering questions, holding review sessions, grading homework and assignments, checking for plagiarism, etc. Most significantly, I (along with the other TA) was responsible for designing the programming assignments, on topics such as basic kernel programming, user-level threads, file systems, etc.

I learned from this experience that undergraduate students have substantially varied skill levels and backgrounds (in terms of the courses taken, coding experience, etc.). To tailor to this diversity, the instructor must carefully evaluate students’ backgrounds and calibrate the assignments; otherwise, there can be a feeling of panic among the students. I experienced this situation firsthand after distributing the basic kernel programming assignment, which I thought had standard instructions, but which students not very familiar with C (but proficient in Java) found difficult to follow.

Mentoring. I have had the opportunity to informally mentor a senior undergraduate (at NYU) and a junior PhD student (at UT Austin). I guided the undergraduate student on how to implement and logically reason about the architecture of a client-side video player for the Popcorn project (a private Netflix-like video service) [2]. Similarly, I introduced the PhD student to research in the area of systems and privacy, and guided him in implementing and evaluating parts of the Pretzel project (a private email service). In particular, I helped him appreciate the significance of an extensive experimental evaluation for a systems paper, and how to go about executing one.

I learned from these experiences that a mentor can improve a mentee’s productivity by tailoring tasks to the likes and strengths of the mentee (e.g., mix literature reading with coding tasks for a student that enjoys coding).

Specifics of my teaching approach and style
After carefully analyzing the aforementioned experiences (what worked and what didn’t work) plus other experiences (student, conference presenter, and researcher), I believe that
my teaching will have the following characteristics in terms of the content and its delivery.

Syllabus

- **Focus on a few topics: cover important foundational material and recent research.** Often I have taken courses where the instructor has covered a broad set of topics. Unfortunately, such a strategy makes it hard for the students to internalize concepts. Instead, I find the style where the instructor goes deep into a few carefully selected and fundamental topics much more effective (for example, my distributed computing and cryptography courses were the ones that followed this strategy and taught me the most). It is also important to cover some recent developments from research and industry to give students the sense that what they are learning is highly applicable.

- **Include pre-lecture readings, programming assignments, and technical writing tasks** because repeatedly going over the same concepts is often required to make students “get it”. Personally, whenever I read about a lecture’s topic beforehand, I felt much more engaged in the classroom, which made the lecture easier to understand. Similarly, I often internalized concepts only while implementing or writing about them.

Content delivery

- **Use the classical style of writing and explaining concepts on the board (rather than using presentation slides),** for various reasons. First, people absorb content at different paces, while slideshow content tends to be delivered at a fast pace. Second, “slide-sized bits” are arbitrary, and force division of related content; instead, a good board removes the artificial slide boundary and allows students to easily cross-reference related content. And, third, boardwork makes students take notes, which helps reinforce the material.

- **Use a (slightly) aggressive style of cold-calling on students** because it keeps everyone on their toes (otherwise, only a few selected students participate and learn) [1], encourages critical thinking, instills the hard skill of being able to effectively communicate technical content in a short time, and reduces the fear of public speaking.

Make it fun! Perhaps the most important part of teaching is to make the experience fun for the students. My physics teacher even played guitar during lectures (although I can’t and will have to figure out something else).

Courses I can teach: I would enjoy teaching both introductory and advanced courses on operating systems and security. Some examples include Introduction to Operating Systems, Advanced Operating Systems, Principles of Computer Systems, Introduction to Security, and Theory and Practice of Secure Systems. In addition, I would enjoy teaching special topics courses in systems, security, and privacy.

References
