SEE TEXT ONLY

TEACH BSD

By Benedict Reuschling and George Neville-Neil

The second week of August 2016, George Neville-Neil visited the Computer Science department at the University of Applied Sciences, Darmstadt, Germany, where I work. George has worked with Robert Watson at the University of Cambridge developing a master-level course over the last several years.

The course uses DTrace to analyze and explore a live FreeBSD operating system. I had arranged for George to come to my university to teach an undergraduate version of the course to a group of interested students in the inter-semester holidays. This endeavour had many goals: we were curious how the course material, available online at teachbsd.org, would work for undergraduate students, especially those who had not taken operating systems courses before. Another goal was to adopt the course to contain less paper writing, which is emphasized in the masters version of the course. while still focusing on practical content. Lastly, we wanted to see how well the German students would understand a course taught in English and whether they would be willing to ask questions and participate in class. We were pleasantly surprised to find that English was not a barrier for the students and they soon started asking questions.

The course was divided into morning and afternoon sessions with a break for lunch. The morning sessions were typical lectures where George explained how different kernel subsystems work, giving both the theoretical background necessary to understand the concepts as well as an overview of the implementation in a modern operating system, FreeBSD. After lunch, the students were given lab assignments to solve using DTrace on virtual machines. The labs covered the topics of what had been taught during the morning lectures. Due to the fact that these guest lectures were scheduled over the semester holidays, we had only a handful of students participating. The small class size created a situation where we were able to give more individual help to students, throughout the week, which is usually not possible with a larger group.

Monday started with an overview of the course content, including a bit of computer history. The introduction and overview sections introduce students to the origins of operating system development, its evolution, and reasons for some of current OS implementations. Many parts of modern operating systems come from the interaction between hardware and software advances, with one influencing the other. One clear example is that modern RISC processors were influenced by the rise of UNIX and the C language; this type of tension motivates many of the lessons we presented to the students. The goal of the lab on Monday was to get the students set up with their virtual machines and to familiarize them with DTrace as a tool, letting them get their hands dirty and take their first, basic, traces.

Tuesday morning started with a short quiz to determine the students' current knowledge level and to gauge how well Monday's topics were understood. We then continued by explaining how processes work, what virtual memory is, and how processes and threads are scheduled in the operating system. A lot of attention was paid to the topic of locking, and what challenges it poses, including deadlocks and lock order reversals. Without a clear understanding of concurrency and locking, students cannot really appreciate many of the subtleties of modern hardware and software, whether in an operating system or in a multi-threaded user program. The lab covered tracing processes and their various states, getting the students to explain the UNIX process life cycle. Feedback from the second day of the course was turned into upated slides. More than once, the night after a class was spent revising or creating whole new sets of slides to illustrate concepts that students struggled to understand during the day.

With the basics of processes, virtual memory, and concurrency having been covered, we then moved on to communication in the operating system. Starting with basic Interprocess Communication, we explained signals and how they are used by programs and the operating system to communicate. From basic IPC we moved on to sockets and network communication. Networking topics, including DNS, UDP, and TCP, were covered with an emphasis on how the operating system implements TCP—connection setup, data transfer, sliding windows, and connection tear down. Since all of the students had already completed an undergraduate networking course, they were able to quickly appreciate what the operating system needed to do to implement correct TCP software. Due to this familiarity, the questions the students asked led deeper into the operating system than in previous sections. The communication lab for that day had students trace the TCP connection setup process using a local nginx instance serving a static webpage.
We also had them visualize their derived state machine using the graphviz package, which I taught the students how to use.

The topic of Thursday's lecture was data storage and filesystems. Things like the namecache, virtual filesystem layer (VFS) were explained, as well as reading and writing data. The namecache always requires a good deal of explanation as the concept of caching negative results is one that most undergraduates have not yet been exposed to. In the lab session, the students explored the name cache a bit more and how it interacts with nginx when a certain web page URL is requested. The class day ended a bit early on Thursday afternoon to give students time to study for the exam on the next day.

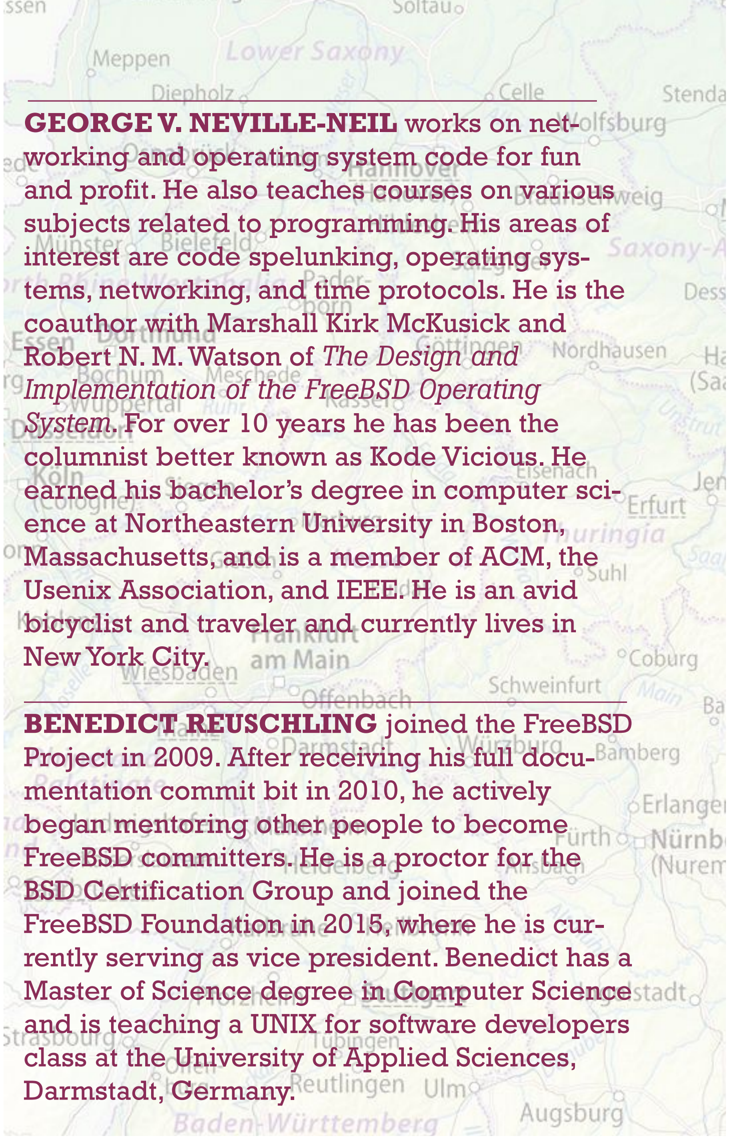
On Friday, we gave the students a whole systems view, combining all the topics discussed during the week. The lecture served not only as a review session before the written exam, but also tied everything together for the students. Specifically, we explored what happens in all the operating system layers when a web server, such as nginx, serves a page. The students were exposed to which worker processes were run, what cache lookups are done, the TCP states that are being created, etc.

The final was a 90-minute exam covering all the topics of the week, with questions focusing on details about the operating system implementation and the writing of a few one-liners for DTrace. Students were allowed to use their laptops throughout the exam in order to test and verify their DTrace scripts before submitting the final versions on their papers. The exam determined the grade for the students in the course. We were very satisfied with the results after grading the exams and we are already discussing how to further enhance the course based on this first test-run.

When George was not teaching (which he enjoyed very much), I was showing him around our campus in Darmstadt, the city, and what good restaurants we have. I also introduced him to some professors who were not on holiday during that week. George not only does great networking on the systems level, but also in person. We had some good discussions, ranging from teaching in computer science to cultural diversity in education and the integration of practitioners' work into the university curriculum.

Some of the more amusing moments of the course related to the inability of all the clocks at the university to keep proper time. Either the batteries were dead, or the timepieces were advancing too slowly or made no sense at all ("nowhere in the world is it 16:00 hours now"). For a timekeeping nerd like George, this proved to be a recurring joke throughout the week.

Overall, it was a very productive week. We gained valuable feedback for the teachbsd course for undergraduates and the students got a lively lecture about operating system internals. I hope to integrate parts of these lectures into my own full-semester course, and George is now planning a return to Darmstadt to teach an updated, and extended, version of the course. ●



GEORGE V. NEVILLE-NEIL works on networking and operating system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, networking, and time protocols. He is the coauthor with Marshall Kirk McKusick and Robert N. M. Watson of *The Design and Implementation of the FreeBSD Operating System*. For over 10 years he has been the columnist better known as Kode Vicious. He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler and currently lives in New York City.

BENEDICT REUSCHLING joined the FreeBSD Project in 2009. After receiving his full documentation commit bit in 2010, he actively began mentoring other people to become FreeBSD committers. He is a proctor for the BSD Certification Group and joined the FreeBSD Foundation in 2015, where he is currently serving as vice president. Benedict has a Master of Science degree in Computer Science and is teaching a UNIX for software developers class at the University of Applied Sciences, Darmstadt, Germany.