Andrew S. Tanenbaum: The Impact of MINIX

Charles Severance, *University of Michigan*

Andrew S. Tanenbaum describes the motivation, development, and market impact of the MINIX operating system.

The story of the evolution of Unix and Unix-like operating systems is very convoluted. Modern Unix-like OSs generally derive from AT&T/BSD Unix or some variation of Linux. But in the beginning, there was only AT&T Unix, and if you trace the moment where the Linux-like OSs flickered to life, you'll find the MINIX OS.

MINIX's original purpose was simply to support university courses that attempted to teach OSes. Essentially, it was a set of floppy disks that came with Andrew Tanenbaum's popular book, *Operating Systems: Design and Implementation*, first published in 1987. I visited him at Vrije Universiteit (VU), and we talked about the motivation, development, and impact of that simple decision to write an OS to teach a course. Visit computer.org/computingconversations to watch our discussion.

EARLY DAYS

AT&T initially provided the source code to its Unix OS to universities. It didn't have an open source license, but it was readily available, which made it a natural choice when teaching courses about OSes. At first, Tanenbaum used a booklet written by John Lions (*John Lions' Commentary on Unix 6th ed., with Source Code*; Peer-to-Peer Communications, 1976) to teach his classes:

> I was teaching a course on operating systems using version 6 Unix; John Lions wrote a book describing it line by line that was very popular at universities. The bean counters at AT&T didn't like this: having every computer science student in the world learn about its product was a horrible idea. Version 7 came out with a license that said, "thou shalt not teach," so you couldn't teach version 7 Unix anymore. It has to be up there as one of the dumbest mistakes in all of business history.

As an accomplished computer scientist who had been teaching OSes for many years, Tanenbaum knew that writing a simple Unix-like OS wouldn't be all that difficult:

> I decided to write my own operating system that was compatible with V7 so that I could teach a course on a Unix-like operating system. I sat down and began writing it, and I got it almost to the point where it worked, but it kept crashing at random. The functionality was there, but it would run for 15 minutes and then crash.

Tanenbaum knew he was close, but he had trouble solving that "one last bug":

> I ran my operating system on a simulator for the PC that was entirely deterministic and reproducible, and then after it crashed, I ran it again and looked at the last 100,000 instructions to find out what happened. It took me a little while to complete the simulator and get it running. If I had been smart, I might have invented VMWare in the 1980s because I had done much of the work but my focus was on Unix.

His OS worked perfectly on the simulator—it could run forever and never crash. Seemingly, it only crashed on the hardware. He was about to give up when he got a lucky break:

> I was confused until one of my students, Robbert van Renesse, mentioned to me that when the Intel CPU chip gets hot, it gives interrupt 15. I wasn't catching interrupt 15 because the manual didn't mention it. I put in a piece of code to catch it, and sure enough,

Published by the IEEE Computer Society

within an hour of running it, I got, "Hi, I'm interrupt 15—this is impossible, and you'll never see this message."

WRITING A NEW BOOK

Once MINIX was working, Tanenbaum wanted to document it by writing a book similar to the one Lions wrote, something that explained MINIX in great detail so that it could be used to teach operating systems courses:

> Most chapters have three sections. The first is the general principles of a component, such as a file system. The second is how these general principles apply to MINIX—how the file system is organized and its key data structures. The third part of every chapter narrates the source code.

Even though this was before the Internet, news of the new book spread rapidly. At one point, a bookstore invited him to give a talk that was so well attended it had to use the Santa Clara Convention Center as the venue. A worldwide bestseller, *Operating Systems: Design and Implementation,* is now in its third edition.

LINUX

Linus Torvalds was a student who picked up MINIX and started evolving the source code:

> Linus went out and specifically bought a PC for the purpose of running MINIX. He was in the MINIX Usenet news group and spent a lot of time developing Linux based on MINIX.

In the early days, there seemed to be no reason to view MINIX or Linux as commercial products. Both were academic exercises, while BSD Unix had 20 years of development and production experience. But AT&T wasn't yet done making strategic errors:

> Kirk McKusick and others who wrote BSD formed a company called Berkeley Software Design Incorporated [BSDi] to sell BSD Unix commercially. Their phone number was 1-800-ITS-Unix. In another brilliant commercial move, AT&T sued BSDi to get rid of it. AT&T had no knowledge at all about how to sell Unix, so it should have gone to the BSDi guys and said, "We want to buy your company—how much?" And it would have had successes: there would have been no MS-DOS, no Windows, and the world would have been all Unix. But instead, AT&T tried to stop these guys, which blocked BSDi from going commercial for several years due to this lawsuit.

While AT&T's lawsuit effectively stalled progress on commercializing BSD Unix, other OSes including Linux began to establish their positions in the market. With all this confusion and with Linux addressing the market for an open source Unix-like system, Tanenbaum kept MINIX's focus on education.

A NEW FUTURE

As the field has evolved, Tanenbaum sees that there might be some value in a highly reliable OS:

> The EU decided to set up a couple of programs where it could give a large amount of money (about €2.5 million) to specific investigators and let them do their thing. I applied for this and got an ERC Advanced grant. With this funding, we're trying to go push MINIX, for embedded systems as well as the x86, especially in the area of high reliability.

Because MINIX is based on a very small kernel with most of the OS functionality in user space processes, it's possible to recover from partial failures or even upgrade OS components on the fly without taking the entire system down:

> If you're running some real-time system like a radio telescope or an electric power plant, you might not want to go down for upgrades, so a system that can update or repair itself while maintaining fairly high availability might have a niche. We've got it working in the lab now for BeagleBoard, BeagleBone, and BeagleBone Black and are about to release them as open source.

Pull Quote here

Regardless of MINIX's possible commercial potential, there's no question that when Andrew created MINIX, he set the wheels in motion for an OS market that isn't the exclusive domain of proprietary vendors.

According to chaos theory, sometimes when a butterfly flaps its wings in Brazil it can cause a hurricane in Florida. If Robbert had not casually mentioned interrupt 15, there would have been no MINIX and no Linux and thus no Android (which is based on Linux). The relative stock prices of Apple and Samsung might be quite different now.

Charles Severance, *Computing Conversations column editor and Computer's multimedia editor, is a clinical associate professor and teaches in the School of Information at the University of Michigan. Follow him on Twitter @drchuck or contact him at csev@umich.edu.*