EDITORIAL

Musings

RIK FARROW

Rik is the editor of *;login:*. rik@usenix.org

When I decided to work with computers, I resolved to help make computers easier for people to use. I had already witnessed through my university classes just how difficult and actually inscrutable computers were, and so I hoped that I could make things better.

I failed. I was waylaid by the usual factor: peer pressure. I wanted to be liked and admired by my peer group, and they were programmers and engineers. We loved coming up with elegant solutions, whether it was in the hidden underpinnings of products or in the user interface.

I wrote one of my first Musings about this in 1998 [1]. In that column, I describe the magic of state machines, beloved of programmers and anathema for just about anybody else. My friends and I would wonder why people couldn't program VCRs or set digital watches when we could figure them out without resorting to manuals!

Things today are different. Instead of state machines, we have graphical interfaces with ever-changing sets of symbology. Three vertically arranged dots sometimes means, "Here's that menu you've been searching for!" but sometimes just leads you off on a wild goose chase instead. You are supposed to learn how your new smartphone works from members of your peer group. And by the time you've figured out how to answer your phone, the interface has been updated and you no longer know how to answer your phone.

The desktop metaphor could be called the visual-spatial interface, as it builds on skills familiar to our ancient ancestors. We locate the icon on the screen and manipulate it using a pointing device. Apple made much out of this interface in the '80s, with Microsoft embracing it in the mid-'90s. Visual-spatial design works well because we are familiar with seeing and pointing.

Consider the modern, touchscreen interface as a counter-example. Instead of pointing to what we want others to notice, imagine that the number of fingers we used while pointing was terribly significant, as was the direction we swiped our pointing fingers afterward. Yes, the two-fingered swipe to the left means "Danger, lion!" Or was that just one finger, meaning, "Food item, attack!" Somehow, I am not surprised that finger gestures never caught on with our not very distant ancestors.

The Lineup

Keeping with my theme of making things easier, more efficient, and definitely cooler, we have gg. Fouladi et al., from Stanford University, have created a suite of tools for converting tasks such as large compilations, running tests, and video processing into thousands of cloud functions. While even the concept of a lambda is certain to bewilder mere mortals, I believe this project will prove a godsend to many of the people who read *;login:*. For anyone using lambdas, I strongly recommend you read Hellerstein et al., the fifth cite in this article.

Jangda et al. built Browsix, a browser extension that extends more complete access to the operating system for applications written in WebAssembly (Wasm). Their purpose was to be able to run standard benchmarking tools, and they have done that and noticed that the performance they get from Wasm is not quite what was promised. Reading this article will teach you more about Wasm, but don't go installing Browsix on the browser you use for everyday tasks.

John Koh, Jason Nieh, and Steve Bellovin created E3, a tool for encrypting email while it is stored on mail servers. Instead of relying on knowledge that Johnny doesn't have and wouldn't understand anyway [2], they have built an interface for adding public-private key pairs and transparently encrypting and decrypting mail messages.

Periwinkle Doerfler is researching the intersection between our apps, the device they run on, and our interpersonal relations. She spoke on this at Enigma 2019 [3], where I met her at lunch and decided I should dig deeper by interviewing her. It shouldn't surprise any of us that our inscrutable devices can be used to further abuse by intimate partners, parents, coworkers, and even others we barely know at all.

Dave Dittrich has been in the trenches, reverse engineering malware and DDoS agents since the late 1990s. More recently, Dave has ventured into policy realms as co-author of the Menlo Report. I borrowed from one of Dave's early projects, and basked in my 15 minutes of fame, when I predicted attacks against the Internet giants of the year 2000 days before the attack began. Dave never stopped, creating, for example, the first Forensic Challenge [4].

Anwar et al., from IBM at Almaden, explain why the manner in which Docker creates containers is inefficient. Docker makes creating container images easy—perhaps too easy, leading to bloated images, wasted storage, and slower startup times with much duplication between layers. They explain why and suggest solutions.

Laura Nolan examined complexity in her previous SRE column and takes on reliability this time. We all want our software to be reliable, and Laura explains some of the key features for building reliable software and provides a detailed checklist you can use to help you and your team do so.

Peter Norton tells us that it's past time to move on to Python 3. Then Peter explains a way to add type checking to Python, both why (if you don't already know) and how it can be done in Python 3.

Dave Josephsen shares some tricks he learned about anomaly detection from Monitorama, and explains how you can use Prometheus's query language (PromQL) to do this yourselves.

Mac McEniry decided it was time for us to learn how to access databases with SQL interfaces from within Go programs. As usual, you can do this fairly simply by using preexisting Go modules, but you still need to understand SQL.

Dan Geer and Brian Wade consider the question: are your Internet-facing hosts more secure on-premises or in the cloud? Using data acquired from a vendor, they provide an intriguing answer.

Robert G. Ferrell considers layers. Applications are layered over libraries and the operating system, and the network consists of some number of layers—just how many and what you name them depends on how you slice things.

Mark Lamourine has written reviews of an older book about continuous delivery and two books on deep learning. I review Neal Stephenson's *Fall*.

I really don't intend to come across like a Luddite. I just hope to remind people who write user-interfacing code that your users will likely not be members of your peer group. Instead, they may be average people interested, even anxious, to partake in the wonderful technology you have created. Perhaps now is the time for a newer, more natural, interface metaphor, or your potential users may be using just one middle finger with which to salute your newest creation.

References

[1] R. Farrow, Musings, *;login:*, vol. 24, no. 4, August 1998, pp. 59–61: http://rikfarrow.com/farrow_aug98.pdf.

[2] A. Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," in *Proceedings of the 8th USENIX Security Symposium (USENIX Security '99)*, USENIX Association, pp. 169–184: https://people.eecs.berkeley.edu/~tygar/papers/Why_Johnny_Cant_Encrypt/USENIX.pdf.

[3] P. Doerfler, "Something You Have and Someone You Know—Designing for Interpersonal Security," Engima 2019: https://www.usenix.org/conference/enigma2019/presentation/doerfler.

[4] The Forensic Challenge: http://old.honeynet.org/challenge/index.html.

Save the Dates!

FAST '20

18th USENIX Conference on File and Storage Technologies

February 24–27, 2020 | Santa Clara, CA, USA

Sponsored by USENIX in cooperation with ACM SIGOPS

Co-located with NSDI '20

www.usenix.org/fast20

The 18th USENIX Conference on File and Storage Technologies (FAST '20) brings together storage-system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems.

The program committee will interpret "storage systems" broadly; papers on low-level storage devices, distributed storage systems, and information management are all of interest. The conference will consist of technical presentations including refereed papers, Work-in-Progress (WiP) reports, poster sessions, and tutorials. Paper submissions are due Thursday, September 26, 2019.

nsdi '20

17th USENIX Symposium on Networked Systems Design and Implementation

February 25–27, 2020 | Santa Clara, CA, USA

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

Co-located with FAST '20

www.usenix.org/nsdi20

NSDI will focus on the design principles, implementation, and practical evaluation of networked and distributed systems. Our goal is to bring together researchers from across the networking and systems community to foster a broad approach to addressing overlapping research challenges.

NSDI provides a high-quality, single-track forum for presenting results and discussing ideas that further the knowledge and understanding of the networked systems community as a whole, continue a significant research dialog, or push the architectural boundaries of network services. Fall paper titles and abstracts are due Thursday, September 12, 2019.

The full program and registration will be available in December.