Better Systems Software

- JEFF FOSTER
- MICHAEL HICKS

Jeff Foster and Michael Hicks are developing ways to help programmers write better systems software—software that is more secure, reliable, and able to run without crashing or needing to be restarted.

Systems software—like the code running workplace servers or Internet shopping sites—must handle many activities in tandem, manage resources like memory and processing power efficiently, and provide ready access to information while also protecting data from unwanted access and tampering.

Hicks and Foster are working to design software that can do all this and also run for years without crashing or needing to be restarted.

Foster and Hicks joined UMIACS within months of each other, both having worked previously on analyzing code for security problems. They continued that work at UMIACS and started collaborating soon after they arrived. In late 2006, they formally merged their research groups. “About half of our students are officially co-advised,” says Hicks, “and the other half are unofficially co-advised.” Everyone in the group meets

Jeff Foster (l), Michael Hicks (r)

Better Systems Software

- JEFF FOSTER
- MICHAEL HICKS

To contact any researcher in UMIACS, go to **www.umiacs.umd.edu**.

frequently to go over their projects and problems.

Their research is both formal and mathematical as well as experimental and practical, the two researchers say. "We build programs that relate to existing software today so that we know that the work that we're doing formally and theoretically will also have practical benefits," says Foster.

One of the problems Hicks and Foster have undertaken is how to update software while it continues to run. For example, an online service or a personal computer could have bugs corrected or get software updates without having to be shut down and restarted. The researchers have designed a flexible and safe system for converting source code into a version that can continue to operate while it is altered. On test runs, Foster and Hicks have shown that common applications, like secure shell servers, FTP servers, or routers could be altered to run the most up-to-date code without having to be shut down at all for three years or more.

"What's very impressive is that the work is based on very strong theory," comments Tom Ball of Microsoft Research, "and they took it all the way to robust implementation."

Another example of a tool that Foster and Hicks have built is a program they call Locksmith. For decades, computers became faster and faster as, true to Moore's law, hardware developers packed ever more transistors into computer processors. But in recent years, the speed of individual processors has plateaued as they have reached the physical limits for transistor density. Packing more transistors together makes processors run too hot and use too much power. As a solution, computers are being made with two processors instead of just one.

"Very soon, we will have computers with hundreds of processors and then thousands," says Hicks. The use of parallel processors fundamentally changes how programmers have to write code, say Hicks and Foster. Software run on parallel processors must be able to integrate calculations that happen separately and make sure that information is transferred between—or among—processors in the proper order. When information isn't processed in the right order, "data races" can cause catastrophic system failures. Indeed, the August 2003 electric blackout in the Northeast United States resulted from a data race in a power plant's management software.

Locksmith can analyze software to ensure that it uses proper synchronization techniques for avoiding data races. This reduces the burden on programmers who are writing for parallel processors since mistakes in synchronization can be caught automatically. Because Locksmith is open-source software, other researchers and developers can freely download, use, and even extend Locksmith.

Another tool that Hicks and Foster have developed, which they call Pistachio, is a method for testing communication protocols among computers to ensure that the protocols are reliable and secure. "There's always a gap between the code that you actually write and the idealized version of it on paper," says Foster. "The idea is to take English-language descriptions of what code is designed to do and translate that into Pistachio's specification language. Then you can run Pistachio to check whether the specifications and the code match." Pistachio runs very quickly and has been used to check several implementations of SSH2, a widely-used protocol for securely communicating between computers, such as between a home computer and an office network.

"Our interest started as technical, but we also want to have an impact," says Hicks about their programs. To increase the usefulness of their tools, he and Foster are working with the Human-Computer Interaction Lab at UMIACS to see how adapting their programs' interfaces could make their tools more accessible and effective. Microsoft's Ball commends the way Hicks and Foster use real data to show that their techniques work. "They have quite a strong track record," he says.

— Profile written by Karin Jegalian

University of Maryland Institute for Advanced Computer Studies
2119 A.V. Williams Building • University of Maryland • College Park, MD 20742

UNIVERSITY OF MARYLAND