Seeking the next Einstein

SHOW ME THE CODE

The openness of free software spurs innovation across generations and time zones. **BY JON 'MADDOG' HALL**

I was probably always subtly aware of the abilities of some free software programmers, so I should not continue to be amazed by what they can do. But I must admit they do continue to astonish me.

For example, Nick lived a few miles from my house and started programming at the age of 9, hacking the Linux kernel at 12, and writing device drivers at 15. He was the senior systems administrator of a small college at 19, and he helped the United States FBI capture some crackers by creating a honeypot at college when he was 21. He then went on to do research – without ever officially graduating from high school.

Or there is the 14-year-old who started his own distribution and had released 20,000 copies before his parents found out what he was doing. When they asked why he did not tell them, he simply answered, "Well, I did not really need your help."

A family friend who was not doing so well in junior high school – armed with a copy of *Running Linux* and an early Linux CD – managed to install Linux to his system, set up a wireless network in his house (including Samba support for his parent's systems), and teach himself C programming. He also formed the first computer club at his high school, came out of his shell, joined the high school football team, later taught himself computer security, and is now doing graduate-level work as an undergraduate.

The Source

All of these hackers credited visual access to the source code as an important factor in improving their computer skills. With free software, few people ask your age, your sex or sexual persuasion, your religion, or anything other than, "Where is the code?" New programmers can work as far and as fast as they want by reading other peoples' code – both good and bad – and learning from it.

In the mid 1970s there was a college professor who believed that the best way to teach programmers how to write good code was to show them the code of really good programmers. John Lions managed to comment and annotate the complete listing of the *Sixth Edition of Unix* before AT&T changed the licensing with the *Seventh Edition of Unix*, which prohibited using the source code for educational purposes. Fortunately for computer science, a few copies "escaped," and that two-volume set of books became on of the most photocopied computer science books of all time.

Unix programmers measured their time in the computer field by whether they owned a fifth-generation photocopy of Lion's book or just a tenth-generation photocopy.

Looking at a good programmer's code is still a great way to learn the craft. I really do not understand how a computer science instructor can advocate using closed source proprietary software to teach students when comparable free software is available.

With proprietary software, you see what the program does, but not how it does it. You must trust that the programmers did the right thing when they chose the algorithms for the program. On the other hand, by teaching how the software does the job, you create an on-going education.

And free software doesn't just show you the code – you can also get to know the programmer. As someone who has worked on both closed source and free software, I appreciate the fact that, if I want to know who wrote a particularly great program and how that person fits into a development environment, I can usually seek out and follow the project's mailing list. This strategy allows me to locate a lot of the really good programmers who are rising to the top of their profession.

This thought leads me to my latest find and the inspiration for this article. I recently learned about an undergraduate student who is participating in what I consider to be graduate-level work. He is articulate, he seems to have a wide range of interests, and he lives in Romania. Free software's openness allowed me to find him, and I hope to work with him in the future as I have worked – and continue to work – with the prodigious coders I mentioned earlier.

I continue to look for the next "Albert Einstein of computer science," and I am not so egotistical to think that this genius would have to come from the United States, or even be educated here. With too many problems to solve and not enough people to solve them, the next "Albert" – or "Alberta" – might be from Brazil, China, Romania, or even Helsinki, Finland.

What I do know is that free software's openness will help us find the next generation of experts. ■