The Disambiguator

Learning about Operating Systems

SELENA DECKELMANN

Selena Deckelmann is a major contributor to PostgreSQL and a data architect at Mozilla. She's been involved with free and open source software since 1995 and began running conferences for PostgreSQL in 2007. In 2012, she founded PyLadiesPDX, a portland chapter of PyLadies. Deckelmann founded Open Source Bridge and Postgres Open, and she speaks internationally about open source, databases ,and community. She is an advisor to the Ada Initiative, an organization dedicated to increasing the participation of women in open source and technology communities. You can find her on twitter (@selenamarie) and on her blog at chesnok.com. selena@chesnok.com

You want to learn about operating systems, and C appears to be the language used in examples you've seen. Now you wonder, "Do I need to learn C?"

My short answer to this is: No.

You don't need to write an operating system from scratch to learn about it. Most of the problems system administrators solve have nothing to do with C, even though many operating systems are written in C. Important concepts such as variables, flow control, loops, arrays, and input/output, are nearly the same in any language. Understand these ideas in one language, and you're on solid ground to learn them in another.

Learning about operating systems is really learning about resource management. Ask yourself questions about your own computer:

- Are the processes you think should be running actually running?
- How much RAM do your processes use over the course of a day?
- How fast are your disks?
- When will you need to upgrade your disks for speed or size?
- How do you apply what you learn to many systems, rather than just your laptop?

Answering these questions with programs you write yourself in shell scripts, Perl, Python, or Ruby will help you learn what you need to know. A book such as the UNIX and Linux System Administration Handbook [1] would also be a helpful guide.

I recently dropped into the classroom of Chris Bartlo, a high school computer science teacher. His kids learn HTML and CSS, Scratch, C++, and Java in their first three years. Now Bartlo is introducing a Python course because he saw first hand how productive a line of Python is when his kids had to solve a series of text parsing problems for a contest.

Bartlo's students can be productive in so many languages because they are in class every day, solving problems. And it's fun—they design and make games, they work in teams, and they plan out their work before they ever write a line of code.

The best way to learn a new language is to have a friend, mentor, or team learning it with you. Pick something your friends use, and you'll learn faster and have more fun.

For those of you picking up another programming language, you can find out what languages are popular on GitHub and Stack Overflow [2]; however, this is an imperfect measure. GitHub and Stack Overflow don't necessarily represent the majority of developers. These developers, though, are probably the trendsetters [3].

So, if you want to learn more about operating systems, start asking and answering questions about them. Use whatever programming language and environment works for you. For some, that could be C. But for me, Python works just fine.

References

[1] Nemeth, Evi, et al. UNIX and Linux System Administration Handbook. Prentice Hall, 2010. Print.

[2] The RedMonk Programming Language Rankings: January 2013: http://redmonk.com/sogrady/2013/02/28/language-rankings-1-13/.

[3] Developer as (Fashion) Designer: http://www.saturnflyer.com/articles/2009/04/28/developer-as-fashion-designer/.