I’m in Brian Kernighan’s Advanced Programming Techniques class at Princeton (COS 333). He’s a brilliant guy with an absurdly rich history in the field of programming. One of my favorite aspects of his lectures is that he constantly drops in little anecdotes from throughout his career (From today’s class: “Oh, I remember when I got Bjarne [Stroustrup] to let me override the comma operator in C++; I’m probably the only person that’s ever used that feature” (there are some juicier tales involving duping governmental figures of significant authority and limited computer skills, but such stories best not leave the classroom)).

In particular, I love hearing his impressions on the development of programming languages and paradigms. He’s been here since (almost) the start of it all, and his opinions are fantastic.

One that particularly stuck with me (from a few weeks back) was when he brought up what’s known as the Sapir-Whorf hypothesis, which roughly suggests that language shapes the way we think and what we think about (disclaimer: I am no linguistics expert).

Kernighan applied this idea to programming languages, suggesting that they influence the way we try to solve problems. And although we can ‘fake’ certain paradigms in any given programming language, it just wouldn’t be natural to do so.

For example, he noted that early Fortran didn’t support recursion; you could “fake it” if you really wanted to (I’m assuming, by converting from a tail-recursive solution to an imperative loop), but it just wouldn’t be an intuitive first approach. Similarly, C doesn’t really provide great object support; you can ‘fake’ objects if you’re so-inclined, but if you had never used OOP before, the language wouldn’t inspire you to take such an approach.

This really resonated with me. I see posts on HN all the time about how ‘C is a functional language’ or ‘C is an OO language’—but in truth, it’s not. We’ve just made it so now that our problem-solving skills have adapted to functional/OOP solutions.