Nicole P. Satiembre BS Computer Engineering-Intern February 20, 2024

Cebu Technological University-Main Campus

97 Things Every Programmer Should Know
Chapters 41-45

41. Interprocess Communication Affects Application Response Time - Randy Stafford

People, especially in this generation, have a very short attention span. I admit, I am one and I am very impatient that I want applications to load within the first 5 seconds, yes, that's how short my patience is, add more the influence of social media in quick gratification. That is why response time is critical to software usability. This chapter teaches me that when we're designing an application, we should be mindful of the number of interprocess communications in response to each stimulus. The passage provides insights into the critical role of response time in software usability, particularly in modern multitier enterprise applications.

When performance is a problem in such applications, examining data structures and algorithms isn't the right place to look for improvements. Response time depends most strongly on the number of remote interprocess communications (IPCs) conducted in response to a stimulus. Each remote interprocess communication contributes some nonnegligible latency to the overall response time, and these individual contributions add up, especially when they are incurred in sequence. For example, many CSS files mean many HTTP requests and this adds delay to the response time. One important lesson is to optimize our program aiming to reduce remote interprocesses.

42. Keep the Build Clean - Johannes Brodwall

In summary, the author advocates for a proactive and disciplined approach to handling compiler warnings, highlighting the benefits of a clean build for code hygiene, team collaboration, and overall software maintainability. Ensuring a consistently clean build eliminates the need for me to repeatedly assess the relevance of warnings. Disregarding warnings requires mental effort, and I aim to minimize unnecessary cognitive load. Maintaining a clean build also facilitates a smoother transition for others taking over my work. If warnings are left unaddressed, someone else would have to navigate through distinguishing between pertinent and non-essential issues. Alternatively, they might opt to ignore all warnings, including those that could be crucial. Warnings from your build are useful. You just need to get rid of the noise to
start noticing them. Don’t wait for a big cleanup.

43. Know How to Use Command-Line Tools - Carroll Robinson

IDEs are a helpful tool for programmers because they are easy to use and they relieve the programmer of thinking about a lot of little details involving the build process. However, it comes with a downside. We tend to rely so much on this tool because it works like magic without needing to know the background process, because of this, we may never fully understand what our tools are actually doing. We will understand better how our tools are doing by working with command-line build tools. Writing our own make files will help us understand all of the steps (compiling, assembling, linking, etc.) that goes into building an executable file. Aside from improving our understanding of the build process, there are also some tasks that can be performed more easily or more efficiently with command-line tools than with an IDE. Command-line tools also support scripting, which allows for the automation of tasks such as producing scheduled daily build options.

This chapter does not intend to discourage us from using IDE. We are encouraged to explore other options and not just limit ourselves to the comfort that IDEs provide. The best way to do that is to learn to use command-line tools. Then when we go back to using our IDEs, we have a better in-depth understanding of the internal mechanism of how our tools work.

44. Know Well More Than Two Programming Languages - Russel Winder

I have learned in this chapter that we shouldn't settle down on one programming language. Sometimes, we tend to become too comfortable in our language of choice that we do not want to deal with learning another language anyway. However, this practice constraints our thinking patterns only to that programming language. A programmer who learns a second language, especially if it is from a different paradigm widens our thinking ability in approaching problems differently. We think about different algorithm implementations and patterns to solve problems better. One paradigm might be better for solving a certain problem than another. It is the idioms of use, not just the syntax and computational models that are the important factors. It is advisable that Programmers should always be interested in learning new languages, preferably from an unfamiliar paradigm, this way, our mind will be more open to different algorithms and approaches to different problems.

45. Know Your IDE - Heinz Kabutz

IDEs are very helpful innovations that make a programmer's life easier. However, it is only helpful if we know the ins and outs of using it. The Integrated Development Environment (IDE) combined the previous editing features with a compiler, debugger, pretty printer, and other tools. Another amazing feature of modern IDEs is the ability to enforce style rules within a company. Personally, I use VS Code because of its flexibility where you can install additional extensions anytime and it is light-weight. I also love using the Prettier extension because it helps in formatting the code at compile time. According to the author, his first step in learning an IDE is to memorize the keyboard shortcuts. Since his fingers are on the keyboard when he is typing his code, pressing Ctrl+Shift+I to inline a variable prevents breaking the flow, whereas switching to navigate a menu with my mouse interrupts it. Aside from that, it is also great to learn how to type fast on your keyboard, it is worth all the time invested up front.