Should .NET developers *really* be spending time learning C for low-level exposure? [closed]

Asked 16 years, 3 months ago Modified 14 years, 5 months ago

Viewed 2k times

14

Closed. This question needs details or clarity. It is not currently accepting answers.

Want to improve this question? Add details and clarify the problem by editing this post.

Closed 4 years ago.

Improve this question

When Joel Spolsky and Jeff Atwood began the disagreement in their podcast over whether programmers should learn C, regardless of their industry and platform of delivery, it sparkled quite an explosive debate within the developer community that probably still rages amongst certain groups today. I have been reading a number of passages from a number of programmer bloggers with their take on the matter. The arguments from both sides certainly carry weight, both what I did not find is a perspective that is uniquely angled from the

standpoint of **developers focused on just the .NET Framework**. Practically all of them were commenting on a general programmer standpoint.

What am I trying to get at? Recall Jeff Atwood's opinion that most of the time developers at such high levels would spend would be on learning the *business/domain*, on top of whatever is needed to learn the technologies to achieve those domain requirements. In my working experience that is a very accurate description of the work life of many. Now supposing that .NET developers can fork the time for "extra curricular" learning, should *that* be C?

For the record, I have learnt C back in school myself, and I can absolutely understand and appreciate what the proponents are reasoning for. But, when thinking things through, I personally feel .NET developers should not dive straight into C. Because, the thing I wish more developers would take some time to learn is - **MSIL** and **CLR**.

Maybe I am stuck with the an unusual bunch of colleagues, I don't know, but it seems to me many people do not keep a *conscious awareness* that their C# or VB code compiles in IL first before JIT comes in and makes it raw machine code. Most do not know IL, and have no interest in how *exactly* the CLR handles the code they write. Reading Jeffrey Richter's **CLR via C#** was quite a shocker for me in so many areas; glad I read it despite colleagues dismissing it as "too low level". I am no expert

in IL but with knowledge of the basics, I found myself following his text easier as I was already familiar with the stack behaviour of IL. I find myself disassembling assemblies to have a look at how the IL turns out when I write certain code.

I learn the CLR and MSIL because I know that is the direct layer *below me*. The layer that allows me to carry out my own layer of work. C, is actually further down. Closer to our "reality" is the CLR and MSIL. That is why I would recommend others to have a go at those, because I do not see enough folks delving at that layer. Or, is your team already all conversant with MSIL?

c clr il

Share

Improve this question

Follow

edited Sep 24, 2008 at 13:57

blowdart

56.5k ● 12 ● 118 ● 151

asked Sep 24, 2008 at 7:15

icelava

9,857 ● 7 ● 53 ● 75

15 Answers

Sorted by: Highest score (default)

36

Of course you should. The greatest way to become overly specialized and single-minded (and, correspondingly, have limited marketable skills) is to only work with a single type of language and eschew all others as "not related to your current task."

Every programmer should have some experience with a modern JIT'd OO language (C#/Java), a lower-level simpler language (C, FORTRAN, etc), a very high level interpreted language (Python, Ruby, etc), and a functional language (Scheme, Lisp, Haskell, etc). Even if you don't use all of them on a day-to-day basis, the broadening of your thought process that such knowledge grants is quite useful.

Share Improve this answer

Follow

answered Sep 24, 2008 at 7:19

Dark Shikari

8,009 ● 4 ● 28 ● 38

1 I really like this answer but you should maybe flip the paragraphs, because it comes off sounding like you think everyone should know `C`. Nonetheless I agree a lot in the fact that having experience all the types of languages is key to being a great developer and not just a good one. I can't recall how many times I've mentioned at work that something would be better done in language x, and people just stare at me blankly. – msarchet Jul 13, 2010 at 13:24

I already know C and that helped me during the 1.1 days where there are a lot of things that are not yet in the .NET

6

base libraries and I have to P/Invoke something from the Platform SDK.

My take is that we should always allocate a time for learning something that we don't know yet. To answer your question, I don't think it is essential for you to learn C but if you have some time to spare, C is a good language to learn and is just as valid as any other language out there.

Share Improve this answer

Follow

answered Sep 24, 2008 at 7:50

jop

84k ● 10 ● 57 ● 52

I think this answer refers to the Win32 API rather than the C programming language. – MarkJ Oct 27, 2009 at 17:55

@Markj i think you need to remember this is asked from the practical standpoint of .NET developers who delve in Windows. – icelava Apr 4, 2010 at 3:51

5

True, C is way below the chain. Knowing MSIL can help devs understand how to optimise their apps better. As for learning C or MSIL, why not both? :)

Share Improve this answer

Follow

answered Sep 24, 2008 at 7:19

cruizer

6,151 ● 2 ● 29 ● 34

Remember, it is about the among of time available. As high-level app developers we spend plenty of time on business/domain matters. – icelava Sep 24, 2008 at 7:29

4

.NET developers should learn about the CLR. But they should also learn C. I don't see how anybody can really understand how the CLR works without some low-level understanding of what happens on the bare metal.

Spending time learning about higher-level concepts is certainly beneficial, but if you concentrate too much on the high-level at the expense of the low-level, you risk becoming one of those "architect" people who can draw boxes and lines on whiteboards but who are unable to write any actual code.

What you learn by learning C will be useful for the remainder of your career. What you learn about the CLR will become obsolete as Microsoft changes their platform.

Share Improve this answer Follow

edited Jul 13, 2010 at 13:18

answered Sep 24, 2008 at 13:56

Kristopher Johnson

82.4k ● 55 ● 251 ● 306

2 The concepts in the CLR (or any other modern VM, such as Java) will certainly outlast the implementation. I don't expect that application developers will ever be going back to an

unmanaged environment in the future - so learning the CLR will be useful for the remainder of your career as well.
– Mark Brackett Jul 13, 2010 at 11:04

3

My take is that learning some compiled language and assembly is a **must**. Without that, you will not get the versatility required to switch between languages and stacks.

To be more specific -- I think that any good/great programmer must know these things by direct experience:

- What is the difference between a register and a variable?
- What is DMA?
- How is a pixel put on the screen (at low level)?
- What are interrupts?
- ...

Knowing these things is the difference between working with a system you **understand** and a system that, for all you know, works by magic. :)

To address some comments

You end up having two different kinds of developers:

- people that can do one thing in 10 ways in one or two languages

- people that can do one thing in one or two ways in 10 different languages

I strongly think that the second group are the better developers overall.

Share Improve this answer Follow

edited Sep 24, 2008 at 8:14

answered Sep 24, 2008 at 7:25

Sklivvz

31.1k ● 24 ● 118 ● 174

1 I appreciate the sentiment, however the advent of cross platform, multilanguage runtimes (like Mono) is really making that a redundant notion. – Jim Burger Sep 24, 2008 at 7:32

Remember I am talking about business/enterprise level application development. Sure there always those special requirements that deal with unique hardware and devices, but most cases it is about interacting with user input, database, web services. Business and customer interaction, not hardware. – icelava Sep 24, 2008 at 7:39

3

I think of it like this:

1. Programmers should probably be actually *working* in the highest-level language appropriate. What's appropriate depends on your scenario. A device driver or embedded system is in a different class from a CRUD desktop app or web page.

2. You want your programmers to have as much practice as possible in the language in which they are working.
3. Since most programmers end up working on generic desktop and web apps, you want programming students to move into the higher level languages as soon as possible during school.
4. However, the higher-level languages obfuscate a few basic programming problems, like pointers. If we apply our principle of using what's appropriate to students as well, those higher level languages may not be appropriate for first year students. That throws out Java, .Net, Python, and many others.
5. So students should use C (or better yet: C++ since it's "higher-level" and covers most of the same concepts) for the first year or two of school to cover basic concepts, but quickly move up to a higher-level language to enable more difficult programs earlier.

edited Sep 24, 2008 at 13:49

answered Sep 24, 2008 at 13:44

Joel Coehoorn
415k ● 114 ● 577 ● 813

1 What type of school? If you mean vocational, then I agree. If you're talking about a CS degree then no, a degree is about the theory and understanding how a computer works, not making it do pretty things. – tloach Sep 24, 2008 at 14:12

3

To be sufficiently advanced in writing C#, you need to understand the concepts in C, even if you don't learn the language proper.

More generally though, if you're serious about **any** skill, you should know what goes on at least one abstraction level below your primary working level.

- Coding in jQuery should be paired with an understanding of JavaScript
- Designing circuits necessitates knowing physics
- Any good basketball player will learn about muscles, bones, and nutrition
- A violinist will learn about the interplay of rosin, friction, bow hairs, string, and wood dryness

Share Improve this answer

Follow

answered Nov 4, 2009 at 19:30

Dinah

54k ● 30 ● 134 ● 149

2

I like to learn a new language every year. Not necessarily to master it, but to force my brain to think in different ways.

I feel learning C is a good language to learn about low level concepts without the pain of coding in assembly.

However I feel that learning lessons from languages like Haskell, python, and even arguably regex (not exactly a language, but you catch my drift?) is as important as the lessons to be gleaned from C.

So I say, learn about the CLR and MSIL on the job if thats your area, and in your spare time, try picking up a different language once every so often. If that happens to be C this year, good for you and enjoy playing with pointers ;)

Share Improve this answer

Follow

answered Sep 24, 2008 at 7:28

Jim Burger

4,537 ● 1 ● 26 ● 27

I used my spare time to learn IL/CLR. I worked overtime to learn other technology suites directly required for the job. :-D – icelava Sep 24, 2008 at 10:57

:-D However it works for you! The important thing is that the good programmers make the spare time to learn in the first place. – Jim Burger Sep 24, 2008 at 12:50

2

I don't see any reason why they should. Languages like Java and C# were designed so that you needn't worry about the low-level details. That's the same like asking whether a WinForms developer should spend time

learning the Win32 API because that's whats happening underneath.

While it doesn't hurt to learn it, you'd probably gain more from spending more time learning the languages and platforms you are familiar with, unless there's a good need to learn the low-level technical details.

Share Improve this answer

Follow

answered Sep 24, 2008 at 17:15

ilitirit

16.3k ● 18 ● 77 ● 115

Is question is given X amount of time available, should you spend that time on MSIL/CLR or C? – icelava Sep 25, 2008 at 1:26

1

It can't be a bad idea to learn MSIL, but in a way it's just another .NET language, but with nasty syntax. It is another layer down, though, and I think people should have at least some vague understanding of all the layers.

C, being somewhat like assembly language with nicer syntax, is a nice way to get an idea of what's happening on quite a low level (although some things are still hidden from you).

And from the other end, I think everyone should know a bit of something like Haskell or Lisp to get an idea of higher-level stuff (and see some of the ideas being introduced in C# 3 in a cleaner form)

answered Sep 24, 2008 at 7:24

Khoth
13.3k • 3 • 30 • 27

If you consider yourself a programmer, I would say yes, learn C.

Many people who write code do not consider themselves programmers. I write .NET apps maybe 3 hours a day at work, but I don't label myself a "programmer." I do a lot of things that have nothing to do with programming.

If you spend your whole day programming or thinking about programming, and you are going to make your entire career revolve arround programming, then you better be sure you know your stuff. Learning C would probably help build a base of knowledge that would be helpful if you're going to go very deep in programming skills.

With everthing, there are trade-offs. The more languages you learn, and the more time you spend dedicated to technology, the less time you have for learning other skills. For example, would it be better to learn C, or read books on project management? It depends on your goals. You want to be the best programmer EVAR? Learn C. Spend hours and hours writing code and dedicating yourself to the craft. You ever want to manage somebody else instead of coding all day? Use the time you would put into programming and find ways to improve your soft skills.

Share Improve this answer Follow

answered Sep 24, 2008 at 13:54

community wiki
Jim

Indeed. As much as I want to be a pure programmer, I am in practical a developer and consultant. There is a challenge to balance technology and business/domain knowledge. Customers' requirements come first unfortunately. – icelava Sep 24, 2008 at 16:25

1

Should .net developers be learning C? I would say "not necessarily," but we should always be dabbling in some language outside of our professional bailiwick because every language brings with it a new way of thinking about problems. During my professional career as a .net (and before that, VB 2-6) developer, I've written small projects in Pascal, LISP, C, C++, PHP, JavaScript, Ruby, and Python and am currently dabbling in Lua and Perl.

Other than C++, I don't list any of them on my resume because I'm not looking to be a professional in any of them. Instead, I bring back interesting ideas from each of them to use in my .net-based work.

C is interesting in that it really gets you close to the OS, but that's hardly the only level you need to know about to be a good programmer.

Share Improve this answer

Follow

answered Sep 24, 2008 at 17:05

Yes - that Jake.

17.1k ● 15 ● 72 ● 99

0

The **CLR** is a **virtual** machine so if that's all you learn, then you only know what's happening at a virtual level.

Learning **C** will teach you more about the **physical** machine as far as memory usage goes, which as you mention is what the CLR uses underneath. Learning how the CLR works isn't going to give you as much insight into, say, garbage collection, as learning C. With C, you *really* appreciate what's involved in memory management.

Learning **CIL** on the other hand, tells you a bit more about execution in .NET than you would by learning **C**. Still, how **IL** maps to machine language will still be a mystery for the most part so knowing some of the high-level opcodes, like the ones for casting types, isn't that helpful in terms of understanding what's really going on as they're opaque for the most part. Learning **C** and pointers, however, will enlighten you on some of those aspects.

Share Improve this answer

Follow

edited Sep 24, 2008 at 7:25

answered Sep 24, 2008 at 7:19

Mark Cidade

99.8k ● 33 ● 229 ● 237

0

Is the issue learning C or MSIL, or is it more fundamental? I'd say that in general, more developers could stand to learn more about how **computers**, physical or virtual, work. A person can get to be a fairly competent programmer by only understanding a language and API in a box. To take the profession to the next level, I feel that developers really need to understand the whole stack. Not necessarily in detail, but in sufficient generality to help solve problems.

A lot of these skills are being talked about here can be acquired by learning more about compilers and language design. You probably need to learn C to do this (whoops, sneaky), but compiler writing is a great context to learn C in. Steve Yegge talks about this on his blog, and I largely agree with him on this point. My compiler writing course in university was one of the most eye opening courses I've ever taken, and I really wish it had been a 200 level course, instead of a 400 level one.

Share Improve this answer

Follow

answered Sep 24, 2008 at 17:27

christopher_f

1,985 ● 1 ● 13 ● 12

0

I posted this on another thread but it applies here to:

I believe you need a good foundation, but devote most of your time to learning what you will be using.

- Learn enough assembler to add two numbers together and display the result on a console. You'll have a much better understanding of what is actually going on with the computer and it will make sense as to why we use binary/Hex. (this can be done in a day and can be done with debug from cmd.exe).
- Learn enough C to have to allocate some memory and use pointers. A simple linked list is sufficient. (this can be done in a day or two).
- Spend more time learning a language that you are going to use. I would let your interests steer you into which language (C#, Java, Ruby, Python, etc.).

Share Improve this answer

Follow

answered Sep 24, 2008 at 18:07

bruceatk

5,138 ● 2 ● 27 ● 36