The Internet Has a Huge C/C++ Problem and Developers Don't Want to Deal With It **354 More** | **Reply** **Login**

The Internet Has a Huge C/C++ Problem and Developers Don't Want to Deal With It

The Fine Print: The following comments are owned by whoever posted them. We are not responsible for them in any way.

With great power comes great responsibility! (Score:5, Insightful)

by RickSchwartz (4662797) on Thursday November 15, 2018 @11:50AM (#57649370) Journal

Guess what? As clever as Python and Java are, you can't effectively write an entire operating system in them, or a high-performance driver like a graphics card driver in them. You could try, but the result would be bloated and slow and effectively useless. So we have compiler languages like C/C++ that require you to actually be a competent programmer who can write code with proper error checking and error handling. I'm not saying that when you have an entire platoon of p[illegible]ng on parts of the same project (vis-a-vis graphics card driver or OS) that there aren't going to be bugs that crop up, but slapping training wheels onto them isn't necessarily the solution to the problem either.

N[illegible] an[illegible] would have this same problem, and for which there is no substitute for in the highest-performance applications: **assembly language**. Yes, Virginia, we still use assembly language in some places, so far as I know. Then you *really* have to know what you're doing.

Nickname:
Password: 6-1024 characters long
□ Public Terminal
Log In Forgot your password?
Close
Close

Maybe the solution to this problem is to educate and train our programmers more thoroughly and carefully.

Reply to This Share
twitter facebook linkedin
Flag as Inappropriate
3 hidden comments

Re:With great power comes great responsibility! (Score:5, Insightful)

by AmiMoJo (196126) <mojo@nOsPaM.world3.net> on Thursday November 15, 2018 @12:23PM (#57649646) Homepage Journal

Flaws in drivers used to be attractive to malicious actors because drivers ran in the kernel and had immense power. Modern operating systems limit drivers so that the damage they can do is greatly reduced. Also helps with stability because your sound card driver crashing won't bring the whole machine down any more.

That's the key to security. Assume stuff will have bugs, isolate it, sandbox it, put layers of security in and limit the damage. A bug in the network stack should at worst crash the network stack, which while annoying is far less critical than being able to read parts of kernel or process memory.

Reply to This Parent Share
twitter facebook linkedin
Flag as Inappropriate

Re: (Score:2)

by Sigma 7 (266129)

Just make the fucking C compiler give you an error when you go out of bounds.

There's at least one platform where you need to use a pointer to a specific point in memory. Perhaps some old programmer may remember "char far *video_buffer = 0xA000:0000" from Borland C when doing basic graphics. Here, the compiler can't tell if you're going out of bounds, since it doesn't automatically know what you're doing.
Same applies to modern C. There may be the odd situation where you access memory directly (e.g. some memo

Re: (Score:2)
by jythie (914043)
I think the solution would be closer to 'right tool for the job'. The problem with C/C++ is you have to work to make it safe, as opposed to something like Python where you have to work to make it unsafe, at least in terms of the type of vulnerabilities the OP is addressing. I think a lot of the problem comes from C/C++ being used too often for too many types of problems, problems that do not really gain from the amount of control the languages give you but do suffer from needing to take thoughtful explici

Re: (Score:2, Insightful)
by Anonymous Coward
Assuming a halfway decent compiler, you could write a graphics driver or a kernel in Rust with little or no performance penalty versus C. You could probably write one in Go. Many have been written in Ada. And I could name many other languages along the same lines. Those are languages that compile down to machine code and can do static allocation and/or explicit dynamic memory management... with static compile-time type checking and pointer and array safety guaranteed.
Yes, array bounds checks are expensive.

Re: (Score:2)
by Rick Schumann (4662797)
My personal opinion of Python: Since I learned how to code in C and assembler, Python gives me an ice-cream headache with all it's weirdnesses. It's a Heinz-57 of programming languages, taking from all over the place, and some of the syntax just made me want to throw things once I discovered them.
1 hidden comment

Re: (Score:3)
by Hizonner (38491)
Then how come every single project's C style guide requires you to indent it just so?
What's really broken is having the humans relying on whitespace (which they DO), and the compiler relying on punctuation.

White space (Score:2)
by jabberw0k (62554)
Python also has a crack dependency on tabs versus spaces, despite tabs being equivalently set at every 8 spaces as God and DEC (but I repeat myself) intended.

Re: (Score:2)
by Ichijo (607641)
Because way too many people don't understand (or don't care about) the difference between an indentation and an alignment. [dmitryfrank.com]
1 hidden comment

Re: (Score:2)
by sfcat (872532)
ANY language where whitespace has more significance than separating words is fundamentally broken.
Mod up!!!

Re: (Score:2)
by Rick Schumann (4662797)
Yeah, that was one of the weirdnesses that made my eyes roll at first.

Re: (Score:2)
by thegarbz (1787294)
you can't effectively write an entire operating system in them
An entire operating system? No. You can however write most of an operating system in them leaving the use of C and C++ for only the situations where low level hardware access is necessary. Citation: The worlds most popular OS running on a Linux kernel.

Re: (Score:3)
by angel'o'sphere (80593)
You can write an OS in any language. There are actually a few OSes written in Java ...
So we have compiler languages like C/C++ that require you to actually be a competent programmer who can write code with proper error checking and error handling.
Error checking etc. is the same in any language ... you know nothing about programming.
assembly language. Yes, Virginia, we still use assembly language in some places, so far as I know. Then you really have to know what you're doing.
You need to know what you are do

Re: (Score:2)
by HornWumpus (783565)
In the early days of microcomputers OSs came in three grades.
1. An OS.

2. An OS that uses basic as it's command interpreter.
3. An OS written in Basic.

Re: (Score:2)
by Tyler Durden (136036)
You can write an OS in any language. There are actually a few OSes written in Java ...
Fantastic. And how many of those are practical enough that you can use them for your serious day-to-day work? Little if any, I bet. And there's a reason for that.

I seriously doubt you could right one in pure Java though. Especially in dynamic memory allocation code, which requires assigning and returning an explicit address at some point. Sure, even C kernels need to employ another language (assembly) from time-to-time
1 hidden comment

Re: (Score:2)
by gweihir (88907)
In addition, it is pretty much as easy to make web-applications effectively as insecure in a memory-safe language. Just screw up permission checking, for example. Otherwise we would not see the flood of vulnerabilities in things written not in C or C++.
The whole article addresses the wrong problem.

Re: (Score:1)
by AndrewFlagg (753349)
well said. bravo. now could some of the VPs outside of technology and engineering that cry all the time, and who sit on high, should know this, oh by the way their password is GOD or PASSWORD and you don't need to hack a buffer overflow to know that. ahem.. i digress again. C/C++ is just fine. just have a great test development team that 1/2 is allowed to read the code and the other 1/2 don't... just white and black box in a closed network environment please..

Re: (Score:2)
by jittles (1613415)
Guess what? As clever as Python and Java are, you can't effectively write an entire operating system in them, or a high-performance driver like a graphics card driver in them. You could try, but the result would be bloated and slow and effectively useless. So we have compiler languages like C/C++ that require you to actually be a competent programmer who can write code with proper error checking and error handling. I'm not saying that when you have an entire platoon of programmers all working on parts of the same project (vis-a-vis graphics card driver or OS) that there aren't going to be bugs that crop up, but slapping training wheels onto them isn't necessarily the solution to the problem either. Note also another 'language' that would have this same problem, and for which there is no substitute for in the highest-performance applications: **assembly language**. Yes, Virginia, we still use assembly language in some places, so far as I know. Then you *really* have to know what you're doing. Maybe the solution to this problem is to educate and train our programmers more thoroughly and carefully.
I can confirm that we still write certain things in ASM for Intel, AMD, and ARM products at my company. Most is written in C because it is ‘fast enough’. But sometimes even C can be too slow and, in certain applications at work, too high level to use effectively.

If God had intended Man to program, we'd be born with serial I/O ports.