Why aren't students taught to use a debugger? [closed]

Asked 15 years ago Modified 5 years, 3 months ago Viewed 9k times

26

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 12 years ago.

The community reviewed whether to reopen this question 3 years ago and left it closed:

> **Opinion-based** Update the question so it can be answered with facts and citations by editing this post.

There's *a lot* of homework questions here on SO.

I would guess that 90%+ can be solved by stepping through the code in a debugger, and observing program/variable state.

I was never taught to use a debugger. I simply printed and read the GDB manual and stepped through their

examples. When I used Visual Studio for the first time, I remembered thinking, Wow! how much simpler can this be, click to set a breakpoint, mouse over a variable for the value, press a key to step, the immediate window, debug.print, etc...

At any rate, are students "taught" to use a debugger? If not, why not? (Perhaps a better question is, why can't they learn to use a debugger themselves... maybe they need to be *told* that there is such a tool that can help them...)

How long does it take to learn to use a debugger?

debugging

Share

Improve this question

Follow

edited Sep 15, 2012 at 17:08

community wiki
4 revs, 4 users 100%
user113476

1 How long does it take to learn? 6-8 weeks. – Jon B Dec 21, 2009 at 21:51

At my university we were taught to use the debugger, both gdb and gui based. – TM. Dec 21, 2009 at 21:51

2 @Jon B - Really? To what level? – user113476 Dec 21, 2009 at 21:53

6 I am in charge of three interns here, one freshman, one junior, and one senior in their respective CS programs. None of which have used a debugger more than "A little bit in one class". I'm not sure why it is not taught more thoroughly... It is something I am pretty much always using. They usually warm up to it within a month or so. – Kevin Dec 21, 2009 at 21:53

1 Maybe the problem is that what's being taught isn't 'how to be a software engineer' but mostly theory? Some universities seem to require more math classes than classes that actually are related to software development. – Jim L Dec 22, 2009 at 0:55

19 Answers

Sorted by: Highest score (default)

29

I don't think the problem is teaching. Using a modern graphical debugger is not rocket science (at least not for most user-mode programs running on a single computer). The problem is with the attitudes of some people. In order to use a debugger effectively, you should:

- Admit it's **your fault** and `select` isn't broken.
- Have the perseverance to spend a couple nights debugging, without forgetting the previous point.
- There's no specific algorithm to follow. You should guess educatedly and reason effectively from what you see.

Not many non-programmers have these attitudes. At college, I have seen many friends who give up after a relatively short period of time and bring me some code and tell me the computer is doing something wrong. I

usually tell them I trust their computer more than them (and this hurts some feelings, but that's the way it is).

Share Improve this answer Follow

edited Dec 21, 2009 at 22:45

community wiki
2 revs
Mehrdad Afshari

2 Point number 1 is definitely the hardest for some of the "developers" I've met. Meh. – Erik Forbes Dec 21, 2009 at 21:57

7 Yes; I've made good money off of bets like "I bet $5 that GCC isn't broken". After losing the bet to me, most people don't repeat that mistake. – Chris Arguin Dec 21, 2009 at 22:08

2 @Chris good thing you were betting for GCC, not javac in the 1.2/1.3 days, otherwise you might not have made off so well. – Suppressingfire Dec 21, 2009 at 23:15

2 I can't understand how anyone cannot think it is their fault. They have written the code, the compiler works for everyone else, how else could the code not perform correctly? – Callum Rogers Dec 21, 2009 at 23:20

6 Actually, there is an algorithm ... it's called the Scientific Method. You observe, spot something that does not jibe with expectations, form a hypothesis of what might cause it, test your hypothesis, comparce with expectations, lather, rinse repeat. – Peter Rowell Dec 22, 2009 at 0:10

In my high school and university, most of the people in the classes didn't really care about programming at all.

7

Share Improve this answer

Follow

answered Dec 21, 2009 at 21:57

community wiki
Tyler Smith

5

If by students you mean Computer Science students, I think the answer is fairly obvious. The subject matter for courses is generally theory, with the programming language / framework / library there as an aid. The professor can't go very far in depth on a particular tool, since it would take away from time he is teaching networking or systems or whatever. Maybe if there were a course called "Real World Programming" or something like that, they'd cover debuggers, but in general I don't see too much wrong with expecting students to read the language / tool documentation in order to accomplish the coursework.

Share Improve this answer

Follow

answered Dec 21, 2009 at 21:57

community wiki
danben

Debuggers were introduced in my second year Intro to C course, if I recall correctly. Of course the problem most students were struggling with at that point was getting

5

their work to compile, which a debugger will not help with. And once their ten line command line program compiles and then crashes, well, they already have some printfs right there. Fighting to master GDB is overkill.

In my experience, it's fairly rare to actually deal with a code base large enough to make more than a cursory familiarization with a debugger worth the time investment in most Comp. Sci curriculums. The programs are small and the problems you face are more along the lines of figuring out the time-space complexity of your algorithm.

Debuggers become much more valuable on real world projects, where you have a lot of code written by different people at different times to trace through to figure out what keeps frotzing foo before the call to bar().

Share Improve this answer Follow

edited Dec 21, 2009 at 23:04

community wiki
2 revs
user185104

1 sometimes code written by you at different times is "hard" enough to use a debugger (It's like when your staring at the screen and wondering who the crap wrote this!? and then look at the header and see it was all you) – Earlz Dec 21, 2009 at 23:33

@earlz I've played that game a few times. – richo Dec 22, 2009 at 0:35

4

This is a good question to ask to the faculty at your school.

At my university, they gave a very brief example of debugging, then pointed us to the "help" files and the books.

Perhaps they don't teach it because there is sooo much stuff to cover and so little time for the lecturers. The professors aren't going to hold everybody's hand.

Share Improve this answer Follow

answered Dec 21, 2009 at 21:53

community wiki
Thomas Matthews

1 That's too bad. If they aren't teaching debugging then they aren't teaching programming. – Jon B Dec 21, 2009 at 21:56

@Jon B: How much programming should a college be teaching? In the Computer Science program I was in, we generally figured it was up to the student to learn to program (although we did have some introductory courses).
– David Thornley Dec 21, 2009 at 22:14

3 @Jon B: That's because they're not teaching programming. They're teaching Computing Science. – user185104 Dec 21, 2009 at 22:29

@mbarnett Yup. That was absolutely the case in my university 35 years ago, and time hasn't done it any favors. The head of the CS department firmly believed that *anyone*

not pursuing theoretical comp-sci, and instead opting for applied comp-sci (e.g. engineering code solutions for evil corporations, or worse, forming your own) was a sell-out. Not only did he not support it, he actively, repeatedly, squelched it at every turn. If your aspirations weren't to become the next Knuth or Dijkstra (and put his department and his college on the map whilst doing so) you were navel lint. – WhozCraig Mar 13, 2022 at 18:33

Not entirely related, but people need to use debuggers not just for debugging but to understand working code.

4

Share Improve this answer Follow

answered Dec 22, 2009 at 0:46

community wiki
JoelFan

@JoelFan - If you have to use a debugger to understand working code, that code needs to be rewritten. – Stephen C Dec 22, 2009 at 0:54

3 @Stephen C - not necessarily... suppose you are new to a project and it's a huge code base... start it up in the debugger and get a feel for the flow between the different parts of the code... perfectly acceptable! – JoelFan Dec 22, 2009 at 14:53

1 @StephenC - sometimes, code is complex because of external requirements, not because it is poorly written. I'm dealing with such a code base at this point. And before you say, "that means your requirements are too complex", I don't think that's the case here. – Vicky Chijwani Mar 14, 2017 at 13:58

@VickyChijwani Put it this way, if code is too difficult to understand without using a debugger, how do you know you have understood it with the debugger. What you see with a debugger is how the code behaves for a particular scenario. To understand how the code works for all scenarios, you really need to understand the code itself ... by reading the code. If the code is too complicated for that, then you have no way to know it is correct. – Stephen C Mar 14, 2017 at 14:14

@VickyChijwani - *"And before you say, 'that means your requirements are too complex' ... "* - I wouldn't say that. I would say that the *implementation* of the requirements is too complex. That could be because the requirements are complex, or because they were expressed in a complex way, or because they were delivered incrementally (e.g. they "just grew"). But even then, it is *usually* possible to come up with an implementation where the code is readable. And if the code is not readable, then you can usually improve it to make it readable. – Stephen C Mar 14, 2017 at 14:19

3

I'll put in a cautionary note on the other side. I learned to program with Visual Basic and Visual C (mid 80s), and the debuggers were built-in and easy to use. Too easy, in fact... I generally didn't think about how to solve a problem, I just ran it in the debugger and adjusted the behavior. Oh, that variable is one too high... I must have to subtract one here!

It wasn't until I switched to Linux, with the not-quite-as-easy gcc/gdb combo, that I began to appreciate design and thinking about your code first.

I'll admit, I probably go too far the other way now. I use a debugger to analyze stack traces and that's about it. There should be a middle ground between analyzing the problem and stepping through it in a debugger. Certainly people should be shown all the tools available too them.

Share Improve this answer Follow

answered Dec 21, 2009 at 21:55

community wiki
Chris Arguin

3 I don't make a lot of friends when I say it (what with working in a microsoft shop), but I think tools like VS make you a worse developer. Falling back on crutches too often leave certain kinds of thinking unexercised, until they pretty much atrophy. – Matt Briggs Dec 21, 2009 at 22:52

1 @Matt I agree and disagree with the statement. I wouldn't want to say to someone "You'll let your legs atrophy if you drive instead of walk" when they need to go cross country. – Jim L Dec 22, 2009 at 0:58

2

I was taught to use a debugger in college. Not much, late (it should be almost the second thing to teach), but they taught me.

Anyway, it's important to teach to DEBUG, not only to "use a debugger". There are situations where you can't debug with gdb (e.g. try to debug a program running 10 concurrent threads) and you need a different approach, like the old-fashioned printf. I can certainly agree with you

that usually one learn and make use of debug techniques much later that the first time you could use them.

Share Improve this answer Follow

answered Dec 21, 2009 at 21:56

community wiki
Khelben

2

From a practicality standpoint, most likely (due to policy or technical restrictions) **you cannot use a debugger on a production application**. Not using the debugger as too much of a crutch promotes adding the proper amount of logging to your application.

Share Improve this answer Follow

answered Dec 21, 2009 at 21:57

community wiki
Chris Ballance

Is there a reason you can't generate a dump when something goes wrong? If policy prevents you from doing so, then that same policy will prevent you from doing a useful amount of logging. – Anon. Dec 21, 2009 at 21:59

2 Aren't logging and debugging two separate things? Logging is there to facilitate and enhance the debugging?
– user113476 Dec 21, 2009 at 21:59

Another reason for not using debuggers on production apps is that they're generally compiled with optimization, and that

breaks the correspondences that a source-level debugger relies on. – David Thornley Dec 21, 2009 at 22:13

@roygbiv logging exceptions and transactions properly can minimize the need for debugging. closely related in use. – Chris Ballance Dec 21, 2009 at 22:23

Debugging dumps (and mini dumps) is something that happens VERY often on large production systems. In fact the rate of minidumps produced by a given system can be something that indicates the priority of an issue for many systems I've interacted with big and small. :-) – Dave Quick Dec 22, 2009 at 0:15

2

Because there is not text book on debugging, period.

In fact, it is very hard to create a teaching situation where get the incentive to use a debuigger. Typical assignments are too simple to *really* require a debugger. Greg Wilson raised that topic at last year's SUITE workshop, and consensus was it is very hard to get students to use a debugger. You can tell them about debugging, but creating situation where they will actually feel the pain of resorting to the debugger is hard.

Maybe a lecture on game cracking could motive students to step through the game with a debugger? At least, that was why I told myself how to use a debugger as a 12-year old :)

Share Improve this answer edited Dec 22, 2009 at 0:53

Follow

community wiki
2 revs
akuhn

1

A found that there is a lot of negative attitude towards debuggers amongst academia and seasoned systems programmers. I have come up against one quite talented programmer who claimed that "Debuggers don't work, i just use log files." Fair enough, for multi-threaded server apps you must have logging, but there's no denying a debugger is useful for 99% of the code that is not multi-threaded.

In answering your question, yes debuggers should be covered in programming syllabus, but as one of the tools of debugging a program. Tracing and logging are important as well.

Share Improve this answer Follow

answered Dec 21, 2009 at 23:13

community wiki
Igor Zevaka

1

Having recruited at 4 universities (Rensselaer, Purdue, Ohio State, University of Washington) - students that write code for money for incubators associated with their university tend to learn the art of debugging really well because the incubation companies want people to solve problems and do it in fewer hours and invest some time

to teach them to use good debugging techniques. Depending on the sophistication of the particular incubator company they might invest in mentoring patterns and performance to help the student be more productive for them but often debugging is the first investment.

Left to the traditional cs classes the students don't seem to walk away with the same set of skills that help them narrow a problem, manipulate the data while the program/service/page/site/component is running andreally understand the implications of what they've written vs. what they needed to write to make it right.

I went to Rensselaer and I 'learned on my own' because I was paid flat rate for some projects and I wanted to minimize my own time spent on programming - and it was further enforced by working as an intern @Microsoft in 1994 where I got to see how useful an Integrated Dev Environment really was.

Share Improve this answer Follow

answered Dec 22, 2009 at 0:25

community wiki
Dave Quick

1

Wagering a hypothesis based on my experience as a TA and aspiring CS prof:



It would actually confuse the kids who have little to no programming experience more than it would help.

Now first of all: I completely agree that teaching how to use a debugger would be a great thing, but I think the barrier to doing so stems from the greater, systematic problem that software engineering and computer science are not separate majors. Most CS programs will require 2-4 classes where learning to code is the focus. After these, coding ability is required but not the topic of the class.

To my main point: It's very hard to teach something using the guise of "you don't get this now but do it because it'll be useful later." You can try, but I don't think it really works. This as an extension of the idea that people only really learn from doing. Going through the motions but not understanding why is not the same as doing.

I think kids learning to code for the first time aren't going to understand why using a debugger is more effective than inserting print lines. Think about a small to medium sized script you code: would you use the debugger barring odd behavior or some bug you couldn't work out quickly? I wouldn't, seems like it would just slow me down. But, when it comes to maintaining the huge project I work on every day, the debugger is invaluable beyond a doubt. By the time students get to the portion of the curriculum that requires big projects, they're not in a class that focuses on general coding anymore.

And all of this brings me to my awesome idea I think every CS prof should do when teaching how to code: instead of exclusively asking for projects from the kids, every now and then give them a big piece of complex code and ask them to fix the bugs. This would teach them how to use a debugger

Share Improve this answer Follow

edited Sep 23, 2019 at 9:30

community wiki
2 revs, 2 users 95%
John M Naglick

0

In high school we were taught to debug by writing stuff out to the console.

In college, we were taught a mix of that plus using a debugger.

The tools have only gotten easier to use, so I am really not sure why it is not taught.

Share Improve this answer Follow

answered Dec 21, 2009 at 21:52

community wiki
Bryan Batchelder

0

I was taught in my first CS class how to use a debugger. It didn't do me much good to be setting breakpoints and stepping through my code when most of what I wrote was "Hello World!". I pretty much ignored the debugger from that point on until I learned to use GDB in a much more advanced course while working on a "binary bomb" homework assignment.

Since I've been out of school and working I've spent a lot more time using a debugger and learning how useful it can be. I would say that learning to use a debugger includes three things - a need for one, being taught/learning how to use one, and experience knowing how to use one to your advantage.

Also, spending some time learning "echo debugging" can be worthwhile for those situations when a debugger isn't available/necessary. That's just my $0.02.

Share Improve this answer Follow

answered Dec 21, 2009 at 21:56

community wiki
Zann Anderson

0

There are more than a few questions here, to my mind. Here's a few that are asked and a few that I'd infer:

I was taught in BASIC and Pascal initially, usually with an interpreter that made it easier to run the program till

something blew up. We didn't have breakpoints or many of the fancy things there are now for tracing through code, though this would have been from 1983-1994 using a Commodore 64, Watcom BASIC, and Pascal on a Mac.

Even in my later university years, we didn't have a debugger. If our code didn't work, we had print statements or do manual tracing, in terms of time this would have been 1995-1997.

One cavaet with a debugger is that for something like Visual Studio, do you have any idea how long it could take to go through every feature it has for debugging? That could take years in some cases I think. This is without getting into all the build options and other things that it can do that one might use eventually. Another point is that for all the good things that a debugger gives, there is something to be said for how complex things can get,e.g. using a breakpoint in VS there is the call stack, local variables, watch windows, memory, disassembly and other things that one could want to examine while execution is halted.

The basics of using a debugger could be learned in a week or so, I think. However, to get to the point of mastering what a debugger does, how much goes on when code is executing as well as where it is executing as there are multiple places where things can run these days like GPUs to go along with the CPU, would take a lot longer and I'd question how many people have that kind of drive, even in school.

Share Improve this answer Follow

answered Dec 21, 2009 at 23:01

community wiki
JB King

0

You're question is kind of similar to, "Why aren't students taught software testing"? I'm sure the do in some places, but typically Universities/Colleges stick to teaching the 'interesting' theoretical computer science stuff, and tend not to teach the practical tools. Like how if you're taking English in school they teach you how to write, not how to use MS Word (yea I'm sure there are some Word courses, but you get my point).

Share Improve this answer Follow

answered Dec 22, 2009 at 0:39

community wiki
Jeremy Raymond

0

I wasn't taught to use a debugger in my undergraduate degree, because you cannot use a debugger on a deck of punch cards. Even print statements are a luxury if you have a 15 minute turnaround on "jobs", etcetera.

I'm not saying that people should not be taught to use debuggers. Just that it is also important to learn to debug without this aid, because:

1. it will help you understand your code better if you don't have to rely on a debugger, and
2. there are situations where a sophisticated debugger won't be available.

On the latter point, I can also remember debugging a boot prom on an embedded device using a (rather expensive) logic analyzer to capture what was happening on the address / data lines.

Share Improve this answer Follow

edited Dec 22, 2009 at 0:52

community wiki
2 revs
Stephen C

0

The same reason students aren't taught version control, or unit testing, or shell scripting, or text editing, or documentation writing, or even (beyond intro courses) programming languages. The class is about computer science, usually a single concept or family of concepts, not programming. You're expected to learn what you need.

This isn't unique to computer science. My chemistry classes (I also have a chemistry degree) didn't teach me how to use any chemistry lab equipment, either. You learned that by hanging around in the lab and watching

other students and asking the grizzled old profs who hung out there.

Share Improve this answer Follow

answered Dec 22, 2009 at 1:02

community wiki
Ken