# What do you do if you cannot resolve a bug? [closed]

Asked 16 years, 2 months ago  Modified 8 years, 4 months ago

Viewed 13k times

34

**Closed**. This question needs to be more focused. It is not currently accepting answers.

**Want to improve this question?** Update the question so it focuses on one problem only by editing this post.

Closed 8 years ago.

Improve this question

Did you ever had a bug in your code, you could not resolve? I hope I'm not the only one out there, who made this experience ...

There exist some classes of bugs, that are very hard to track down:

- **timing-related bugs** (that occur during inter-process-communication for example)
- **memory-related bugs** (most of you know appropriate examples, I guess !!!)

- **event-related bugs** (hard to debug, because every break point you run into makes your IDE the target for mouse release/focus events ...)
- **OS-dependent bugs**
- **hardware dependent bugs** (occurs on release machine, but not on developer machine)
- ...

To be honest, from time to time I fail to fix such a bug on my own ... After debugging for hours (or sometimes even days) I feel very demoralized.

What do you do in this situation (apart from asking others for help which is not always possible)?

Do you

- use pencil and paper instead of a debugger
- face for another thing and return to this bug later
- ...

Please let me know!

**debugging**



edited Apr 17, 2015 at 10:20

community wiki

4 revs, 2 users 98%
Thomas Koschel

This may have been closed as too broad, but a darn good question, timeless and relevant. – Daniel May 28, 2019 at 21:22

## 32 Answers

Sorted by: Highest score (default)



42

Some things that help:

1) Take a break, approach the bug from a different angle.

2) Get more aggressive with tracing and logging.

3) Have another pair of eyes look at it.

4) A usual last resort is to figure out a way to make the bug irrelevant by changing the fundamental conditions in which it occurs

5) Smash and break things. (Stress relief only!)

Share Improve this answer Follow

answered Sep 30, 2008 at 20:30

community wiki
Matias Nino

Personally, #4 scares me. If you don't know what the bug is, how will you ever know that you have made it irrelevant.
– Torlack Sep 30, 2008 at 21:05

4 It becomes irrelevant when it never happens again. ;)
– Matias Nino Sep 30, 2008 at 23:45

1 Agree with all of these - number 3 is especially important. Even just explaining it to someone can help (see the Cardboard Dog) – Greg Nov 3, 2008 at 11:27

1 There's one more thing that I would add to your summary (between 2 and 3). Temporarily disable bits of code to the point that the bug stops occuring – Christiaan Westerbeek Oct 2, 2018 at 20:54

20

I once worked for a company that sold a client-server application that was basically a file transfer and synchronization tool. Both the client and the server were custom applications we had designed.

We had a persistent bug that was very hard to duplicate in the lab. Our server could only handle a certain number of incoming client connections per box, so many of our customers would "cluster" multiple servers together to handle large user populations. The back end data for the cluster was on a file server they all shared. In this cluster configuration there was a bug that would happen under load where we would get a low-level file system error code on a file sharing call involving one of the back end files. Nobody could get this to repeat reliably in the lab,

and even when they could they couldn't narrow down what was happening.

(I forget the exact error, it was probably `59 ERROR_UNEXP_NET_ERR` or maybe `65 ERROR_NETWORK_ACCESS_DENIED`. As I recall it was not even one of the documented error codes you were supposed to be able to get from the API we were calling, which was usually a lock or unlock call on a file section).

Since it involved the communication between the server and the back-end file store, and I was the "network transport" guy, I was tasked with looking at it. Many others had looked at it with no luck.

The one solid thing I had was I knew where in the code the error was being detected, but not what to do about it. So I needed to find the root cause. So I set up an appropriate hardware environment to duplicate it, and I put a custom build of the server software that instrumented the section of code in question.

The instrumentation was as follows: I added a test for the troublesome error code, and had it call a piece of code to send a UDP packet to a predetermined network address when the error occurred. The UDP packet contained a unique string in it to key on.

I then set a packet sniffing tool on the network. (At the time I was using Microsoft Network Monitor). I positioned it where it would be able to "see" this UDP packet when it

was sent as well as all the communication between the cluster servers and the file server.

Most good sniffers have a mode where you can have it capture until it sees a particular piece of traffic, then stop. I turned on that mode and set it to look for that UDP packet my code would send. The goal was to end up with a packet capture of all the file server traffic right before the bug occurred. The very last network packets to and from the system where the UDP packet originated would presumably be a big clue as to what was happening.

I set the "stress test" configuration going and went home for the weekend.

When I got back on Monday, lo and behold I had my data. The sniffer had stopped just as expected after many hours of running and contained a capture. After studying the capture, what I found was that the Server Message Block or SMB (aka CIFS aka SAMBA) connection between our server and the file server was actually *timing out* at the TCP level due to extreme loading on the server. Because all of Microsoft's stuff is heavily layered, it would percolate back up through the file sharing stack as an "unexpected" error instead of returning a more intelligible error code that said "hey, you lost your connection at the TCP level".

I did a little more research on the TCP settings for Windows, and lo and behold the defaults for the version of Windows we were using (probably NT 4 in that era) were none too generous. It was only allowing for a very

small number of failures on the TCP connection and boom, you were dead. Once you lost your SMB connection to the file server, all your file locks were toast and there was no way to recover.

So I ended up writing an *appendix to the user manual* that explained how to alter the TCP settings in Windows to make your cluster server a bit more tolerant of high load situations. And that was it. The fix to the bug was *zero change in code*, merely some additional documentation on how to properly configure the OS for use by this product.

What have we learned?

- Be prepared to run altered versions of your code to investigate the problem
- Consider using non-traditional tools to solve the problem (sniffers)
- Not all bug fixes require code changes
- Sometimes you can diagnose a bug while at home having a beer



answered Sep 30, 2008 at 21:01

community wiki
Tim Farley

Great comment Tim. This is a great example of what is sometimes required to isolate a difficult bug. – Kluge Sep 30, 2008 at 21:18

I'm actually going through something similar to this right now. Hey, that's why we get paid, right? – Rich Oct 1, 2008 at 6:26

Not all bug fixes require code changes(even some change in configuration setting can fix the bug) – ManirajSS Aug 16, 2015 at 17:49

12

I do a number of different things:

- throw out all my assumptions and start from scratch. Remember, a bug exists because something which appears correct is actually wrong. Even those lines or functions or classes that you are absolutely certain are correct may be incorrect. Until you can convince yourself of the correctness you can't assume anything is right.
- keep putting in print statements and assert statements to eliminate things and allow me to reform new assumptions.
- step through code in the debugger if the problem is a control flow problem. Don't step over functions. Step in them and go through all the detail of their execution to confirm they are working right. Confirm the arguments and return values.
- If a line or function or class is suspect but I can't prove it in situ, then write a small test case that does

what you think the problem construct does. This may locate the problem or give some insights as to where to look next.

- stop for the day. It's amazing what kind of offline processing your brain will do overnight. Often the answer or a key insight will appear the next day while I'm doing something mindless like showering or driving.

Share Improve this answer

Follow

answered Sep 30, 2008 at 20:41

community wiki
mxg

---

9

Create an automated way to cause the bug. The worst bug to fix is one that takes hours to reproduce.

Share Improve this answer

Follow

answered Sep 30, 2008 at 20:35

community wiki
Robert Deml

---

6

Quote taken from "**The Cryptonomicon**":

*"Intuition, like a flash of lightning, lasts only for a second. It generally comes when one is tormented by a difficult decipherment and when one reviews in his mind the*

*fruitless experiments already tried. Suddenly the light breaks through and one finds after a few minutes what previous days of labor were unable to reveal."*

Share Improve this answer Follow

answered Sep 30, 2008 at 20:34

community wiki
arul

---

I usually ask someone else to take a look at the code. While I'm explaining what the code is supposed to do, I sometimes see the bug just as I talk.

6

When a bug is a tough one, I sit and work until I figure it out and solve the problem. Interestingly enough, there are times when catching a mysterious bug is more enjoyable than everything running smoothly. And the relief and feeling when a bug is resolved, well, not many other things can beat that (except the obvious ones).

Share Improve this answer Follow

answered Sep 30, 2008 at 20:37

community wiki
petr k.

Also works with rubber ducks... (and is indeed called rubber duck debugging) – idmean Jul 11, 2017 at 7:05

3

I have definitely had bugs which I worked on for 4-5 days continuously before finding a solution. Other bugs have sat in the bug tracker for months, as I put in a few hours spread out over a long period of time. I think this sort of bug is inevitable in any complex software project.

Some stuff that works well for me:

- binary search through the program flow with logging
- use `Trace` statements along with DbgView to search for bugs which show up in release mode
- find an alternate way to reproduce the bug without changing the code
- (works against logic, but...) change the code so that the bug is more easily reproducible (the failure condition is more readily achieved)
- sleep on it and try again tomorrow with a fresh pair of eyes :)

The worst sort of bug in my opinion is a concurrency bug which disappears when logging is inserted.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:48

community wiki
Nick

If all else fails, don't tackle it directly. Rewrite the problem area code in a more refactored way.

2

Share Improve this answer Follow

answered Sep 30, 2008 at 20:34

community wiki
Brian R. Bondy

---

Lots of great answers here. One thing that's worked for me in the past is to ask "what can I do to make it totally obvious when this problem has occured?".

2

For example, if the problem is a corrupted value in a data structure, try building a consistency-check routine that you can run periodically. Also consider implementing all access to the shared data through a set of functions that log each change.

Or, if the problem is a "random" memory overwrite, use a replacement malloc()/free() implementation that traps writing to "free" memory (like electric fence or dmalloc).

Someone else mentioned automating the process of triggering the bug. This is greeat if you can do it. Even having a routine that randomly exercises the program might help in these cases.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:47

community wiki
Mark Bessey

---

2

Seriously? I do things in this order.

1. Go to bed
2. Ask a colleague
3. Rewrite so the area isn't affected.
4. Ask SO
5. Raise a support ticket with your 3rd party library vendor.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:49

community wiki
Johnno Nolan

---

2

"What do you do in this situation (apart from asking others for help which is not always possible)?"

When is it not possible to ask for help?

There are always others you can turn to for assistance - your coworkers, your boss, friends here at Stack Overflow, etc.

Understanding when to seek help shouldn't be demoralizing!

Share Improve this answer Follow

answered Sep 30, 2008 at 21:01

community wiki
Joe Strazzere

---

2

There are a lot of good tips here.

One that I absolutely do not agree with is the concept of changing the code hoping that it will go away. First off, you a probably going to introduce new bugs. Seconds, you can easily change things enough to hide the bug only to have it resurface again with the next patch.

Memory corruption bugs are especially likely to vanish as magically as they turn up. However, the memory corruption bug isn't fix, it is only that non-fatal areas of memory are getting trashed.

1) Try a different debugger. For example, I use WinDbg more and more often. When you load a program in a debugger, memory layout for your application will change slightly. Maybe a different debugger cause the error to manifest slightly differently.

2) If you resort to changing code without knowing exactly what the problem is, then if the bug goes away, **YOU**

**MUST** go back and understand why the change fixed the bug. Otherwise, you are probably just hiding the bug.

3) Talk to others about the bug, maybe they have seen different versions of the same problem (i.e. other ways to recreate it)

4) Logging.

Share Improve this answer Follow

answered Sep 30, 2008 at 21:03

community wiki
Torlack

---

2

I've had bugs that took weeks or months before a solution was found, but eventually all bugs do get fixed. Aside from the classical non-debugger bug-tracking techniques like disabling parts of the system until you get a minimal test case, I've used these techniques:

- Looking for better debugging tools. A new perspective goes a long way. Xdebug is something I started using in PHP only because of a performance bug that I wasn't making headway on.
- Studying the technology that the bug is located in. This has helped to debug an outlook add-in. It had random errors that made no sense and that google searches turned up zilch about. By researching outlook add-in best practices, COM and MAPI

programming, we got a clearer picture of what could go wrong, and thought of new things to try to fix the bugs, which eventually did fix them.

- Trying to exacerbate the problem. If there's an issue that only happens occasionally, I'll try to find ways to make it happen constantly. This has helped to track down errors in web apps under IE and also to narrow down a crashing bug in the flash plugin.
- When all else failed, I've rewritten the subsystem that caused problems from scratch. This may take a few days, or even weeks, but if you're stuck on a bug, and can't resolve it, and customers won't take no for an answer, what else can you do? This doesn't always fix things, but if it doesn't, you usually get a clearer picture of what's going wrong.

I've noticed a few commonalities in these bugs that I get stuck on for weeks:

- Asking 3rd parties for help rarely helps, and it's generally not a good idea to wait for someone else to come save the day.
- Almost always the fault is inside some 3rd party closed source technology, especially when using obscure parts. IE had nasty bugs when trying to use client certificates. Flash didn't deal well with randomly generated drawing instructions (some of which were nonsensical). Outlook doesn't like it when you try to change form layout dynamically from code.

These days I've learned to respect the "comfort zones" of proprietary tech.

Share Improve this answer Follow

answered Sep 30, 2008 at 21:09

community wiki
Joeri Sebrechts

---

2

I give it more time. I once had a bug (in a personal project) that I just could not figure out. I tried every debugging method I could think of, including Google, with no success. Six months later, I came back and found the bug within an hour or so. It wasn't something simple (something apparently undocumented was going on deep inside Swing), but I just looked at it in a way I hadn't before.

Share Improve this answer Follow

answered Sep 30, 2008 at 21:23

community wiki
Michael Myers

1 +1: I agree with you, but it's difficult to demand more time when management expects you to meet a deadline. And I'll allow that immovable deadlines are a different issue entirely. – Jim G. Jul 25, 2010 at 16:52

1

I've had this problem before, I believe everyone has, I have flat out given up before, it was simply impossible to find, yet it kept crashing, when theres some kind of bug in the code, what I do is just sit down and concentrate on every bit of the code little by little until I find it, it's hard and it takes patience but it's all you can do in such a situation.

Hope this helps.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:30

community wiki
Rayne

---

1

I honestly cannot recall a bug that I couldn't fix. It may cause a lot of refactoring, or may take a while, but I've never had one that I can't get rid of. If it takes me more than an hour to track it down then it's almost always something *really* stupid and small like looking right past that `:` that should've been a `;`, etc.

In python, if I'm using an editor that isn't mine, or maybe it's someone else's code, I use `retab!` in vim, or paste into something like pastie to check indentation (if I don't have vim available).

If it's not a crasher/deal breaker, then I move on and come back with a fresh pair of eyes.

Oh, and you can *never*, *ever* have too much logging.

Share Improve this answer

Follow

answered Sep 30, 2008 at 20:31

community wiki
camflan

---

1

I add as much debug as possible (write to log file, message boxes, etc.), and test.

I don't think this is the worst bug you can find. The worst ones are those you can't reproduce deterministically or in the testing environment.

Share Improve this answer

Follow

answered Sep 30, 2008 at 20:34

community wiki
friol

---

1

I get a bit demoralized too when unable to solve a bug. Usually when I hit a wall with a bug, I would just take note on my findings and stop working on it. I would jump on another bug that is easier to solve and then came back to the bug. By doing this, I would have a fresh mind and attitude in tackling the bug. Sometimes you might have tendency to overcomplicate things when spending too

much times on a bug. Having a break, helps in breaking the wall.

RWendi

Share Improve this answer Follow

answered Sep 30, 2008 at 20:35

community wiki
RWendi

---

1

First off, is it reproducible? That's a HUGE plus if it is. I want things bugs to always/never happen... its the intermittent ones that are the troublesome ones.

And it is going to depend on the problem, but at my shop we'll generally tag-team such a problem figuring that 2 heads (or 3 or 4) is better than 1.

Occasionally the bug won't even be in MY code, but it generally is. There have been issues where a 3rd party library was the culprit or a particular implementation on a particular platform was the cause - those stink.

I'll use anything and everything to at least track it down: debuggers, trace output, whatever.

Typically, if I can isolate it to a class or module I'll write a test harness to duplicate the real world and try to duplicate it there. I generally write my test code first, but

sometimes legacy code (or other developer's code) exists that doesn't have tests already.

I generally will talk the design and problem through, out loud with the team and whiteboard anything that isn't clear. Often the solution will bubble to the surface once we talk about it as a group.

That's what I do.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:37

community wiki
itsmatt

---

1

I usually, try hard solving it. But, if that is not possible for reasonable windows of time, I leave it for some time to braincells to solve it while i sleep ;) Sometime it works...

Share Improve this answer Follow

answered Sep 30, 2008 at 20:37

community wiki
f13o

---

1

I've considered asking for help on this website called StackOverflow that I've been frequenting lately...

Share Improve this answer Follow

edited May 23, 2017 at 11:46

community wiki
2 revs
Adam Bellaire

---

1

This is what I did today...

I debug HW/SW interaction and its often the case logging (instrumentation) changes or hides the bug. Hence tests are performed "at-speed". I call these bugs "roaches" as they run away from any light I can shine on them.

So I have to:

Find the transaction that causes the bug. List the HW interaction via logging (this test passes, but it illustrates the flow).

Instrument before and after the bug to print state changes.

The bug I'm solving now of course is worst case as the HW locks up. The HW includes the CPU so its like being in a well lit room then the power fails and its pitch black.

I have a special backdoor view into memory, but of course this is locked up also. I tried power cycling in the hopes that the memory would stay non-volatile long enough to reenable the backdoor. No such luck. This is possible though.

I very very carefully wrote all the steps I went through to characterize this bug (what works, what fails etc). Sent this to developers with similar HW to verify it just wasn't me or my HW.

I took a few hours break to let this info settle and see if any lightbulbs lit elsewhere.

No replies, this bug is mine to solve...

This HW SW interaction is a loop tha does some setup then enters a polling loop that reads when the transaction is finished. Many transactions should occur. Which transaction fails? Is it the first one (indicating I can debug the transaction and not some noise in the HW). Is it the always the Nth transaction? What makes the Nth different than the first or the (N-1)th. The SW is single threaded and built to be predictable. No preemption, no interrupts enabled.

This SW has worked before, whats new? All the HW is new. In this case all the silicon is new as its an ASIC. Even the embedded CPU is new and customized so the ISA is new.

So I suspect everything and I'm blind. I'll have to sneak up on this roach.

I enabled just the log that reports how many times the SW polls the HW for completion. In this way the first transaction runs at speed, I get an idea how often I touch the HW in a tight polling loop. The test passes. I know its

the Nth transaction and I recorded the peak number of polls for all transactions (perhaps meaningless data).

After modifing anythin, I have to put it back the way it was to verify the bug still exists. After all the earth has rotated and the solar winds are not as strong ;)

Looked at all the checkins, saw a contractor changed some important setup parameters with no explanation. These (outsourced) people are still under evaluation. This will not help.

Found there was no spinwait in the polling loop. Bad for the loop timeout as without it the timeout depends on CPU speed. Added spinwait, still no happiness.

Limited the number of transactions to see where it fails, somewhere before 1000.

Setup the HW to run slower, still hangs.

Hate to leave anyone reading this hanging too, but this diatribe will have to wait till tomorrow.



answered Oct 1, 2008 at 0:19

community wiki
humble_guru

1

**There is no bug that can't be fixed, since there is no bug that can't be fixed with a total rewrite.**

An unfixable bug is just a bug you aren't willing to replace.

Share Improve this answer Follow

answered Oct 1, 2008 at 1:26

community wiki
MusiGenesis

I'd vote myself down for pomposity if I could. – MusiGenesis Oct 1, 2008 at 1:29

0

For memory related bugs i have found that the Memory Profiling options of Ants Profiler have helped me quite a bit on finding bugs.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:30

community wiki
Mitchel Sellers

0

use more creative methods of tracking the bug down.

using remote debugging on the machine where its reproducable.

using profiling tools.

introduce more logging to the app.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:31

community wiki
Andreas Petersson

---

0

Going away for a while and then coming back to a problem is one common approach I do and have heard.

How easily reproduced the bug is can be a factor as well since if the error only occurs in one in a zillion runs of a program that could be considered a negligible gain for fixing it by breaking something else.

There is also the question of nailing down where the bug is, is it in some configuration so that it occurs on a server but not my local XP Pro machine which runs IIS 5.0. Some other bugs may involve having to change the resolution of my machine that can be annoying to try to reproduce a bug that others have reported.

You left out the "occurs under another O/S" category of bugs so that a web page that is fine in IE and Firefox on PC may look like crap on Safari on a Mac. Do I get my hands dirty in trying to fix a CSS issue using my machine as a server and the Mac that is over a row or two in the

cubicles of the floor in order to see this issue or is it so low a priority it gets swept under the rug? Alternatively, if a bug was on Linux and there aren't any Linux machines near me, what should I do?

I'm sorry to have left with some questions but these seem to be difficult questions for me at times.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:35

community wiki
JB King

0

In addition to the debugger, I've also used logging and old fashioned paper and pencil. On occasion I've found really hard bugs, like code that runs fine in debug mode, but breaks in release mode. I've even occasionally rewritten perfectly good code that for whatever reason, doesn't work reliably, figuring that it's better to be reliable than elegant.

I sometimes try to redefine what others term a bug as really being a feature, but that seldom works!

Share Improve this answer Follow

answered Sep 30, 2008 at 20:35

community wiki
Kluge

---

0

I have a bug that shows up every few months on a customer site. It usually happens at 3am and it's not discovered until early the next morning when the customer arrives at their site. And usually when they discover it, they want everything to get working immediately, so our support people generally just reboot the computer. It's been driving me nuts for years. It never happens on my test machine or in the QA lab, only at certain customer sites. Over time, I've

- refactored some of the code that I thought was causing it

- added more debugging printouts around where it appears to be crashing
- redirected stdout so that next time I see it I can "`kill -3`" the process
- given support some new tools to dump out the current state of database locks and the like.
- added diagnostics to make it more obvious when it does happen

It hasn't happened in a few months, and I've got my fingers crossed that I might have fixed it this time, but I'm not counting on it.

Share Improve this answer Follow

answered Sep 30, 2008 at 20:41

community wiki
Paul Tomblin

---

0

If it's not critical, don't fix it, you'll just spend too much time!

Keep the bug open. comment/work on it when you can. It might get fix by accident (or by someone else) later on!

Share Improve this answer Follow

answered Sep 30, 2008 at 20:43

community wiki

elmarco

0

Sometimes it takes a little lateral thinking, but every bug is fixable. Sometimes you need to leave it and sleep over it, sometimes it's good to ask someone else to have a quick look (they may see something you haven't), but mostly it's about trying different things, calling up on previous experience. It can be frustrating, but the buzz you get when you do fix it, is like no other!

Share Improve this answer

Follow

answered Sep 30, 2008 at 20:46

community wiki
Steve