What's the toughest bug you ever found and fixed? [closed]

Asked 16 years, 2 months ago Modified 13 years, 2 months ago

Viewed 54k times

63

votes

What made it hard to find? How did you track it down?

Not close enough to close but see also https://stackoverflow.com/questions/175854/what-is-the-funniest-bug-youve-ever-experienced

debugging

Share

edited May 23, 2017 at 10:31

community wiki
9 revs, 7 users 57%
Martin Beckett

This thread should be required reading for ANY budding programmer. Good question! – Thorsten79 Dec 14, 2008 at 10:33

Comments disabled on deleted / locked posts / reviews

73 Answers

Sorted by: Highest score (default)

1 2 3 Next

200
votes

A jpeg parser, running on a surveillance camera, which crashed every time the company's CEO came into the room.

100% reproducible error.

I kid you not!

This is why:

For you who doesn't know much about JPEG compression - the image is kind of broken down into a matrix of small blocks which then are encoded using magic etc.

The parser choked when the CEO came into the room, because he always had a shirt with a square pattern on it, which triggered some special case of contrast and block boundary algorithms.

Truly classic.

Share

answered Oct 11, 2008 at 12:04

community wiki
sharkin

3 reminds me of this story... "A nursing mother once approached the Rogatchover Rebbe, a brilliant 19th century European Torah scholar, with a perplexing problem...." tfdixie.com/parshat/mishpatim/018.htm – JoelFan Jun 30, 2009 at 5:13

115
votes

This didn't happen to me, but a friend told me about it.

He had to debug a app which would crash very rarely. It would only fail on Wednesdays -- in September -- after the 9th. Yes, 362 days of the year, it was fine, and three days out of the year it would crash immediately.

It would format a date as "Wednesday, September 22 2008", but the buffer was one character too short -- so it would only cause a problem when you had a 2 digit DOM on a day with the longest name in the month with the longest name.

Share

edited Oct 4, 2008 at 13:05

community wiki
2 revs
James Curran

1 Kudos! That is a truly awesome bug – Dave Cheney Oct 9, 2008 at 23:33

Wow. That ones going to my scrapbook. – Mostlyharmless Oct 24, 2008 at 21:35

Wow, that's a hell-of-a-bug! How long did it take for your friend to come across this? – Michael Klement May 5, 2009 at 7:08

5 We had something similar. A buffer on the stack that was one byte short. The date & time were being formatted *without* leading zeroes. Only for months > 9, days > 9, hours > 9, minutes > 9, seconds > 9 would the ending zero byte step on the high byte (this was a 68k) of another value on the stack. And this only mattered if that value didn't already have 0 in it *and* if the routine didn't exit early for other reasons. This was in a home-grown multi-threaded environment. It took over 2 weeks to find it. – Peter Rowell May 13, 2009 at 22:46

60 votes

This requires knowing a bit of Z-8000 assembler, which I'll explain as we go.

I was working on an embedded system (in Z-8000 assembler). A different division of the company was building a different system on the same platform, and had written a library of functions, which I was also using on my project. The bug was that every time I called one function, the program crashed. I checked all my inputs; they were fine. It had to be a bug in the library -- except that the library had been used (and was working fine) in thousands of POS sites across the country.

Now, Z-8000 CPUs have 16 16-bit registers, R0, R1, R2 ...R15, which can also be addressed as 8 32-bit registers, named RR0, RR2, RR4..RR14 etc. The library was written from scratch, refactoring a bunch of older libraries. It was very clean and followed strict programming standards. At the start of each function, every register that would be used in the function was pushed onto the stack to preserve its value. Everything was neat & tidy -- they were perfect.

Nevertheless, I studied the assembler listing for the library, and I noticed something odd about that function --- At the start of the function, it had PUSH RR0 / PUSH RR2 and at the end to had POP RR2 / POP R0. Now, if you didn't follow that, it pushed 4 values on the stack at the start, but only removed 3 of them at the end. That's a recipe for disaster. There an unknown value on the top of the stack where return address needed to be. The function couldn't possibly work.

Except, may I remind you, that it WAS working. It was being called thousands of times a day on thousands of machines. It couldn't possibly NOT work.

After some time debugging (which wasn't easy in assembler on an embedded system with the tools of the mid-1980s), it would always crash on the return, because the bad value was sending it to a random address. Evidently I had to debug the working app, to figure out why it didn't fail.

Well, remember that the library was very good about preserving the values in the registers, so once you put a value into the register, it stayed there. R1 had 0000 in it. It would always have 0000 in it when that function was called. The bug therefore left 0000 on the stack. So when the function returned it would jump to address 0000, which just so happened to be a RET, which would pop the next value (the correct return address) off the stack, and jump to that. The data perfectly masked the bug.

Of course, in my app, I had a different value in R1, so it just crashed....

Share

edited Oct 19, 2011 at 21:03

community wiki
5 revs, 3 users 95%
James Curran

I have to ask -- is location 0000 special on the Z-80000 (a la i86's interrupt vectors) and how is the RET opcode encoded? What a great debug. – Jim Nelson Dec 11, 2008 at 7:11

I don't recall if Z-8000 had interrupt vectors like that. The 8080 and Z-80 did, so it's a possibility. If it did, we weren't using them. – James Curran Dec 12, 2008 at 14:30

How long did it take you to figure it out? – Gad Jun 26, 2009 at 8:59

50
votes

This was on Linux but could have happened on virtually any OS. Now most of you are probably familiar with the BSD socket API. We happily use it year after year, and it works.

We were working on a massively parallel application that would have many sockets open. To test its operation we had a testing team that would open hundreds and sometimes over a thousand connections for data transfer. With the highest channel numbers our application would begin to show weird behavior. Sometimes it just crashed. The other time we got errors that simply could not be true (e.g. accept() returning the same file descriptor on subsequent calls which of course resulted in chaos.)

We could see in the log files that something went wrong, but it was insanely hard to pinpoint. Tests with Rational Purify said nothing was wrong. But something WAS wrong. We worked on this for days and got increasingly frustrated. It was a showblocker because the already negotiated test would cause havoc in the app.

As the error only occured in high load situations, I double-checked everything we did with sockets. We had never tested high load cases in Purify because it was not feasible in such a memory-intensive situation.

Finally (and luckily) I remembered that the massive number of sockets might be a problem with select() which waits for state changes on sockets (may read / may write / error). Sure enough our application began to wreak havoc exactly the moment it reached the socket with descriptor 1025. The problem is that select() works with bit field parameters. The bit fields are filled by macros FD_SET() and friends which DON'T CHECK THEIR PARAMETERS FOR VALIDITY.

So everytime we got over 1024 descriptors (each OS has its own limit, Linux vanilla kernels have 1024, the actual value is defined as FD_SETSIZE), the FD_SET macro would happily overwrite its bit field and write garbage into the next structure in memory.

I replaced all select() calls with poll() which is a well-designed alternative to the arcane select() call, and high load situations have never been a problem everafter. We were lucky because all socket handling was in one framework class where 15 minutes of work could solve the problem. It would have been a lot worse if select() calls had been sprinkled all over of the code.

Lessons learned:

- even if an API function is 25 years old and everybody uses it, it can have dark corners you don't know yet
- unchecked memory writes in API macros are EVIL
- a debugging tool like Purify can't help with all situations, especially when a lot of memory is used

- Always have a framework for your application if possible. Using it not only increases portability but also helps in case of API bugs
- many applications use select() without thinking about the socket limit. So I'm pretty sure you can cause bugs in a LOT of popular software by simply using many many sockets. Thankfully, most applications will never have more than 1024 sockets.
- Instead of having a secure API, OS developers like to put the blame on the developer. The Linux select() man page says

> "The behavior of these macros is undefined if a descriptor value is less than zero or greater than or equal to FD_SETSIZE, which is normally at least equal to the maximum number of descriptors supported by the system."

That's misleading. Linux can open more than 1024 sockets. And the behavior is absolutely well defined: Using unexpected values will ruin the application running. Instead of making the macros resilient to illegal values, the developers simply overwrite other structures. FD_SET is implemented as inline assembly(!) in the linux headers and will evaluate to a single assembler write instruction. Not the slightest bounds checking happening anywhere.

To test your own application, you can artificially inflate the number of descriptors used by programmatically opening

FD_SETSIZE files or sockets directly after main() and then running your application.

Thorsten79

Share

edited Oct 5, 2008 at 9:02

community wiki
8 revs
Thorsten79

Hard to believe this had no up-votes before I got here... good lesson! – Rich Oct 5, 2008 at 8:35

These days epoll is an even better solution to this problem – bdonlan May 19, 2009 at 1:19

43
votes

Mine was a hardware problem...

Back in the day, I used a DEC VaxStation with a big 21" CRT monitor. We moved to a lab in our new building, and installed two VaxStations in opposite corners of the room. Upon power-up,my monitor flickered like a disco (yeah, it was the 80's), but the other monitor didn't.

Okay, swap the monitors. The other monitor (now connected to my VaxStation) flickered, and my former monitor (moved across the room) didn't.

I remembered that CRT-based monitors were susceptable to magnetic fields. In fact, they were -very- susceptable to

60 Hz alternating magnetic fields. I immediately suspected that something in my work area was generating a 60 Hz alterating magnetic field.

At first, I suspected something in my work area. Unfortunately, the monitor still flickered, even when all other equipment was turned off and unplugged. At that point, I began to suspect something in the building.

To test this theory, we converted the VaxStation and its 85 lb monitor into a portable system. We placed the entire system on a rollaround cart, and connected it to a 100 foot orange construction extension cord. The plan was to use this setup as a portable field strength meter,in order to locate the offending piece of equipment.

Rolling the monitor around confused us totally. The monitor flickered in exactly one half of the room, but not the other side. The room was in the shape of a square, with doors in opposite corners, and the monitor flickered on one side of a diagnal line connecting the doors, but not on the other side. The room was surrounded on all four sides by hallways. We pushed the monitor out into the hallways, and the flickering stopped. In fact, we discovered that the flicker only occurred in one triangular-shaped half of the room, and nowhere else.

After a period of total confusion, I remembered that the room had a two-way ceiling lighting system, with light switches at each door. At that moment, I realized what was wrong.

I moved the monitor into the half of the room with the problem, and turned the ceiling lights off. The flicker stopped. When I turned the lights on, the flicker resumed. Turning the lights on or off from either light switch, turned the flicker on or off within half of the room.

The problem was caused by somebody cutting corners when they wired the ceiling lights. When wiring up a two-way switch on a lighting circuit, you run a pair of wires between the SPDT switch contacts, and a single wire from the common on one switch, through the lights, and over to the common on the other switch.

Normally, these wires are bundeled together. They leave as a group from one switchbox, run to the overhead ceiling fixture, and on to the other box. The key idea, is that all of the current-carrying wires are bundeled together.

When the building was wired, the single wire between the switches and the light was routed through the ceiling, but the wires travelling between the switches were routed through the walls.

If all of the wires ran close and parallel to each other, then the magnetic field generated by the current in one wire was cancelled out by the magnetic field generated by the equal and opposite current in a nearby wire. Unfortunately, the way that the lights were actually wired meant that one half of the room was basically inside a large, single-turn transformer primary. When the lights were on, the current flowed in a loop, and the poor monitor was basically sitting inside of a large electromagnet.

Moral of the story: The hot and neutral lines in your AC power wiring are next to each other for a good reason.

Now, all I had to do was to explain to management why they had to rewire part of their new building...

Share

edited Sep 27, 2010 at 20:42

community wiki
2 revs, 2 users 98%
mkClark

13 Debugging the building? Is it even programming-related? +1 :) – ya23 May 13, 2009 at 22:43

---

34 votes

A bug where you come across some code, and after studying it you conclude, "There's no way this could have ever worked!" and suddenly it stops working though it always did work before.

Share

answered Oct 4, 2008 at 4:30

community wiki
David L

I see that all the time, very odd. – stimms Oct 4, 2008 at 5:04

5 Also known as a Schroedinbug. Not sure if that qualifies as "tough" though. – finnw Oct 4, 2008 at 10:55

I agree -- I've seen it time and again over the last 20 years.
– Phil Bennett Oct 4, 2008 at 13:51

5 this happens a lot, but usually it's just that once you realize it, you are more careful with the repro steps or the input data.
– moogs Nov 17, 2008 at 12:17

6 Tends to happen to me the other way around. The code doesn't work, and then I step through it and step through it and conclude to myself that it MUST work. And then it does.
– Nick Lewis Jul 24, 2009 at 22:13

---

28 votes

One of the products I helped build at my work was running on a customer site for several months, collecting and happily recording each event it received to a SQL Server database. It ran very well for about 6 months, collecting about 35 million records or so.

Then one day our customer asked us why the database hadn't updated for almost two weeks. Upon further investigation we found that the database connection that was doing the inserts had failed to return from the ODBC call. Thankfully the thread that does the recording was separated from the rest of the threads, allowing everything but the recording thread to continue functioning correctly for almost two weeks!

We tried for several weeks on end to reproduce the problem on any machine other than this one. We never could reproduce the problem. Unfortunately, several of our other products then began to fail in about the same manner, none of which have their database threads separated from the rest of their functionality, causing the entire application to hang, which then had to be restarted by hand each time they crashed.

Weeks of investigation turned into several months and we still had the same symptoms: full ODBC deadlocks in any application that we used a database. By this time our products are riddled with debugging information and ways to determine what went wrong and where, even to the point that some of the products will detect the deadlock, collect information, email us the results, and then restart itself.

While working on the server one day, still collecting debugging information from the applications as they crashed, trying to figure out what was going on, the server BSoD on me. When the server came back online, I opened the minidump in WinDbg to figure out what the offending driver was. I got the file name and traced it back to the actual file. After examining the version information in the file, I figured out it was part of the McAfee anti-virus suite installed on the computer.

We disabled the anti-virus and haven't had a single problem since!!

Share

answered Oct 4, 2008 at 7:30

community wiki
Miquella

23 Urgh - AV products and servers should never come in contact with each other – Dave Cheney Oct 11, 2008 at 7:15

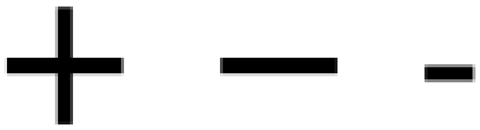
McAfee is terrible.. I didn't know it's THAT evil! – ya23 May 13, 2009 at 22:37

We have similar, completely random problems of almost every variety here, all of which can be traced back to antivirus software. – BlueRaja - Danny Pflughoeft Jan 22, 2010 at 19:08

26 votes

I just want to point out a quite common and nasty bug that can happens in this google-area time:
code pasting and the infamous minus

That is when you **copy paste some code with an minus in it, instead of a regular ASCII character hyphen-minus ('-')**.


Plus, minus(U+2212), Hyphen-Minus(U+002D)

Now, even though the minus is supposedly rendered as longer than the hyphen-minus, on certain editors (or on a DOS shell windows), depending on the charset used, it is actually rendered as a regular '-' hyphen-minus sign.

And... you can spend hours trying to figure why this code does not compile, removing each line one by one, until you find the actual cause!

May be not the toughest bug out there, but frustrating enough ;)

(Thank you ShreevatsaR for spotting the inversion in my original post - see comments)

Share

edited May 23, 2017 at 11:54

community wiki
4 revs
VonC

So, better to code from scratch, perhaps? – Myrrdyn Oct 4, 2008 at 13:22

Well, when you are copy-pasting code, it is precisely with the intention to *not* code from scratch ;) – VonC Oct 4, 2008 at 14:34

6 I recommend a binary rather than a linear search for this sort of thing. – twk Oct 5, 2008 at 5:52

Instead of removing each line one by one, do a binary search. That'll save you some time :-) – Jasper Bekkers May 5, 2009 at 9:43

2 Wow, I had a very similar issue, copy-pasting text from Adobe InDesign to a (plain-text) XML file for use in a web site. Half the time the XML was reported to be malformed due to hyphens and ellipses. Most of the time the invalid characters were obvious... apart from the time it was some *different type of space* character!! Turns out there are at least 5 different space characters, which an "A List Apart" article highlighted nicely for me. – DisgruntledGoat May 13, 2009 at 22:35

19
votes

The first was that our released product exhibited a bug, but when I tried to debug the problem, it didn't occur. I thought this was a "release vs. debug" thing at first -- but even when I compiled the code in release mode, I couldn't reproduce the problem. I went to see if any other developer could reproduce the problem. Nope. After much investigation (producing a mixed assembly code / C code listing) of the program output and stepping through the assembly code of the released product (yuck!), I found the offending line. But the line looked just fine to me! I then had to lookup what the assembly instructions did -- and sure enough the wrong assembly instruction was in the released executable. Then I checked the executable that my build environment produced -- it had the correct assembly instruction. It turned out that the build machine somehow got corrupt and produced bad assembly code for only one instruction for this application. Everything else (including previous versions of our product) produced identical code to other developers machines. After I showed my research to the software manager, we quickly re-built our build machine.

Share

answered Oct 4, 2008 at 5:22

community wiki
Kevin

3 I hope he took you out for drinks and gave you a raise... – Tad Oct 24, 2008 at 21:56

2 Kudos for finding this particularly nasty bug. Maybe ECC RAM could have helped? – Thorsten79 Dec 14, 2008 at 10:31

1 This deserves to be much higher than it is – BlueRaja - Danny Pflughoeft Jan 22, 2010 at 19:12

16 votes

Somewhere deep in the bowels of a networked application was the line (simplified):

```
if (socket = accept() == 0)
    return false;

//code using the socket()
```

What happened when the call succeeded? `socket` was set to 1. What does `send()` do when given a 1? (such as in:

```
send(socket, "mystring", 7);
```

It prints to `stdout`... this I found after 4 hours of wondering why, with all my `printf()`s taken out, my app was printing to the terminal window instead of sending the data over the network.

Share

answered Oct 6, 2008 at 23:27

community wiki
Claudiu

12 gcc -Wall would have caught that – Adam Rosenfield Oct 28, 2008 at 20:53

Laughing my ass off with this one. Too awesome. – karlphillip Dec 3, 2010 at 2:06

15 votes

With FORTRAN on a Data General minicomputer in the 80's we had a case where the compiler caused a constant 1 (one) to be treated as 0 (zero). It happened because some old code was passing a constant of value 1 to a function which declared the variable as a FORTRAN parameter, which meant it was (supposed to be) immutable. Due to a defect in the code we did an assignment to the parameter variable and the compiler gleefully changed the data in the memory location it used for a constant 1 to 0.

Many unrelated functions later we had code that did a compare against the literal value 1 and the test would fail. I remember staring at that code for the longest time in the debugger. I would print out the value of the variable, it would be 1 yet the test 'if (foo .EQ. 1)' would fail. It took me a long time before I thought to ask the debugger to print out what it thought the value of 1 was. It then took a lot of hair pulling to trace back through the code to find when the constant 1 became 0.

Share answered Oct 5, 2008 at 12:40

community wiki
Bryan Oakley

1 We call this the *bad* kind of dynamism
– BlueRaja - Danny Pflughoeft Jan 22, 2010 at 19:13

12 votes

I had a bug in a console game that occurred only after you fought and won a lengthy boss-battle, and then only around 1 time in 5. When it triggered, it would leave the hardware 100% wedged and unable to talk to outside world at all.

It was the shyest bug I've ever encountered; modifying, automating, instrumenting or debugging the boss-battle would hide the bug (and of course I'd have to do 10-20 runs to determine that the bug had hidden).

In the end I found the problem (a cache/DMA/interrupt race thing) by reading the code over and over for 2-3 days.

Share

answered Oct 4, 2008 at 4:22

community wiki
Mike F

1 and after countless hours doing boss battling! +1 – Axarydax Apr 19, 2010 at 9:15

12

Not very tough, but I laughed a lot when it was uncovered.

votes

When I was maintaining a 24/7 order processing system for an online shop, a customer complained that his order was "truncated". He claimed that while the order he placed actually contained N positions, the system accepted much less positions without any warning whatsoever.

After we traced order flow through the system, the following facts were revealed. There was a stored procedure responsible for storing order items in database. It accepted a list of order items as string, which encoded list of `(product-id, quantity, price)` triples like this:

> "<12345, 3, 19.99><56452, 1, 8.99><26586, 2, 12.99>"

Now, the author of stored procedure was too smart to resort to anything like ordinary parsing and looping. So he directly transformed the string into SQL multi-insert statement by replacing `"<"` with `"insert into ... values ("` and `">"` with `");"`. Which was all fine and dandy, if only he didn't store resulting string in a varchar(8000) variable!

What happened is that his `"insert ...; insert ...;"` was truncated at 8000th character and for that particular order the cut was "lucky" enough to happen right between `insert`s, so that *truncated SQL remained syntactically correct*.

Later I found out the author of sp was my boss.

Share answered Oct 7, 2008 at 18:48

community wiki
Constantin

12
votes

This is back when I thought that C++ and digital watches were pretty neat...

I got a reputation for being able to solve difficult memory leaks. Another team had a leak they couldn't track down. They asked me to investigate.

In this case, they were COM objects. In the core of the system was a component that gave out many twisty little COM objects that all looked more or less the same. Each one was handed out to many different clients, each of which was responsible for doing `AddRef()` and `Release()` the same number of times.

There wasn't a way to automatically calculate who had called each `AddRef`, and whether they had `Release`d.

I spent a few days in the debugger, writing down hex addresses on little pieces of paper. My office was covered with them. Finally I found the culprit. The team that asked me for help was very grateful.

The next day I switched to a GC'd language.*

(*Not actually true, but would be a good ending to the story.)

Share

answered Oct 24, 2008 at 21:14

community wiki
Jay Bazuzi

1 I feel with you. Been doing the same for colleagues. My speciality is finding double free()s on global or object variables ;-) – Thorsten79 Dec 14, 2008 at 10:37

12 votes

Bryan Cantrill of Sun Microsystems gave an excellent Google Tech Talk on a bug he tracked down using a tool he helped develop called dtrace.

The The Tech Talk is funny, geeky, informative, and very impressive (and **long**, about 78 minutes).

I won't give any spoilers here on what the bug was but he starts revealing the culprit at around 53:00.

Share

answered Oct 24, 2008 at 22:08

community wiki
Tim Stewart

12 votes

While testing some new functionality that I had recently added to a trading application, I happened to notice that the code to display the results of a certain type of trade would never work properly. After looking at the source control system, it was obvious that this bug had existed for at least a year, and I was amazed that none of the traders had ever spotted it.

After puzzling for a while and checking with a colleague, I fixed the bug and went on testing my new functionality. About 3 minutes later, my phone rang. On the other end of the line was an irate trader who complained that one of his trades wasn’t showing correctly.

Upon further investigation, I realized that the trader had been hit with the exact same bug I had noticed in the code 3 minutes earlier. This bug had been lying around for a year, just waiting for a developer to come along and spot it so that it could strike for real.

This is a good example of a type of bug known as a Schroedinbug. While most of us have heard about these peculiar entities, it is an eerie feeling when you actually encounter one in the wild.

Share

edited Jul 15, 2010 at 9:57

community wiki
2 revs
RoadWarrior

9
votes

The two toughest bugs that come to mind were both in the same type of software, only one was in the web-based version, and one in the windows version.

This product is a floorplan viewer/editor. The web-based version has a flash front-end that loads the data as SVG. Now, this was working fine, only sometimes the browser would hang. Only on a few drawings, and only when you wiggled the mouse over the drawing for a bit. I narrowed the problem down to a single drawing layer, containing 1.5 MB of SVG data. If I took only a subsection of the data, any subsection, the hang didn't occur. Eventually it dawned on me that the problem probably was that there were several different sections in the file that in combination caused the bug. Sure enough, after randomly deleting sections of the layer and testing for the bug, I found the offending combination of drawing statements. I wrote a workaround in the SVG generator, and the bug was fixed without changing a line of actionscript.

In the same product on the windows side, written in Delphi, we had a comparable problem. Here the product takes autocad DXF files, imports them to an internal drawing format, and renders them in a custom drawing engine. This import routine isn't particularly efficient (it uses a lot of substring copying), but it gets the job done. Only in this case it wasn't. A 5 megabyte file generally imports in 20 seconds, but on one file it took 20 minutes, because the memory footprint ballooned to a gigabyte or more. At first it seemed like a typical memory leak, but memory leak tools reported it clean, and manual code inspection turned up nothing either. The problem turned out to be a bug in Delphi 5's memory allocator. In some conditions, which this particular file was duly recreating, it would be prone to severe memory fragmentation. The system would keep trying to allocate large strings, and find nowhere to put them except above the highest allocated memory block. Integrating a new memory allocation library fixed the bug, without changing a line of import code.

Thinking back, the toughest bugs seem to be the ones whose fix involves changing a different part of the system than the one where the problem occurs.

Share

answered Oct 4, 2008 at 13:54

community wiki
Joeri Sebrechts

5 "the toughest bugs seem to be the ones whose fix involves changing a different part of the system than the one where the problem occurs." How very true. Finding the bug in your SW is hardest if the bug is not even in *your* SW... – sleske Jun 24, 2009 at 23:32

9 votes

When the client's pet bunny rabbit gnawed partway through the ethernet cable. Yes. It was bad.

Share

answered Oct 4, 2008 at 15:49

community wiki
slashmais

2 Nope. Honest truth. Re-installed windows twice IIRC .. Did check that the cable was firmly plugged in, etc. It was only by accident that we discovered the gnawed section of the cable - it was about an inch wide on one side only, but cut through some of the internal wires. Main horror was that some of the network functionality worked... – slashmais Dec 22, 2009 at 14:50

8
votes

Had a bug on a platform with a very bad on device debugger. We would get a **crash** on the device **if we added a printf** to the code. It then would crash at a different spot than the location of the printf. If we moved the printf, the **crash would ether move or disappear**. In fact, if we changed that code by reordering some simple statements, the crash would happen some where unrelated to the code we did change.

It turns out there was a **bug in the relocator for our platform.** the relocator was not zero initializing the ZI section but rather using the relocation table to initialze the values. So any time the relocation table changed in the binary the bug would move. So simply added a printf would change the relocation table an there for the bug.

Share

answered Oct 4, 2008 at 5:07

community wiki
Ted

The old TRS-80 had memory initialized with alternating 0x00 and 0xff blocks, each (IIRC) 64 or 256 or whatever bytes long. One program stopped working when the author added a splash screen. It relied on an uninitialized variable that was pushed into the 0xFF area by the three-byte CALL instruction to call the splash screen routine. – David Thornley May 13, 2009 at 22:07

We had the same issue with a boot loader miscalculating sizes etc and print's causing issues because of ordering etc. – David d C e Freitas Sep 15, 2011 at 19:03

8 votes

This happened to me on the time I worked on a computer store.

One customer came one day into shop and tell us that his brand new computer worked fine on evenings and night, but it does not work at all on midday or late morning. The trouble was that mouse pointer does not move at that times.

The first thing we did was changing his mouse by a new one, but the trouble were not fixed. Of course, both mouses worked on store with no fault.

After several tries, we found the trouble was with that particular brand and model of mouse. Customer workstation was close to a very big window, and at midday the mouse was under direct sunlight. Its plastic was so thin that under that circumstances, it became translucent and sunlight prevented optomechanical wheel for working :|

Share

answered May 5, 2009 at 7:03

community wiki
user58196

7
votes

My team inherited a CGI-based, multi-threaded C++ web app. The main platform was Windows; a distant, secondary platform was Solaris with Posix threads. Stability on Solaris was a disaster, for some reason. We had various people who looked at the problem for over a year, off and on (mostly off), while our sales staff successfully pushed the Windows version.

The symptom was pathetic stability: a wide range of system crashes with little rhyme or reason. The app used both Corba and a home-grown protocol. One developer went so far as to remove the entire Corba subsystem as a desperate measure: no luck.

Finally, a senior, original developer wondered aloud about an idea. We looked into it and eventually found the problem: on Solaris, there was a compile-time (or run-time?) parameter to adjust the stack size for the executable. It was set incorrectly: far too small. So, the app was running out of stack and printing stack traces that were total red herrings.

It was a true nightmare.

Lessons learned:

- Brainstorm, brainstorm, brainstorm
- If something is going nuts on a different, neglected platform, it is probably an attribute of the environment platform
- Beware of problems that are transferred from developers who leave the team. If possible, contact the previous people on personal basis to garner info and background. Beg, plead, make a deal. The loss of experience must be minimized at all costs.

Share

answered Oct 4, 2008 at 5:05

community wiki
Michael Easter

7
votes

Adam Liss's message above talking about the project we both worked on, reminded me of a fun bug I had to deal with. Actually, it wasn't a bug, but we'll get to that in a minute.

Executive summary of the app in case you haven't seen Adam message yet: sales-force automation software...on laptops...end of the day they dialed up ...to synchronize with the Mother database.

One user complained that every time he tried to dial in, the application would crash. The customer support folks went through all their usually over-the-phone diagnostic tricks, and they found nothing. So, they had to relent to the ultimate: have the user FedEx the laptop to our offices. (This was a very big deal, as each laptop's local database was customized to the user, so a new laptop had to be prepared, shipped to the user for him to use while we worked on his original, then we had to swap back and have him finally sync the data on first original laptop).

So, when the laptop arrived, it was given to me to figure out the problem. Now, syncing involved hooking up the phone line to the internal modem, going to the "Communication" page of our app, and selecting a phone number from a Drop-down list (with last number used pre-selected). The numbers in the DDL were part of the customization, and were basically, the number of the office, the number of the office prefixed with "+1", the number of the office prefixed with "9,,," in case they were calling from an hotel etc.

So, I click the "COMM" icon, and pressed return. It dialed in, it connected to a modem -- and then immediately crashed. I tired a couple more times. 100% repeatability.

So, a hooked a data scope between the laptop & the phone line, and looked at the data going across the line. It looked rather odd... The oddest part was that I could read it!

The user had apparently wanted to use his laptop to dial into a local BBS system, and so, change the configuration of the app to use the BBS's phone number instead of the company's. Our app was expecting our proprietary binary protocol -- not long streams of ASCII text. Buffers overflowed -- KaBoom!

The fact that a problem dialing in started immediately after he changed the phone number, might give the average user a clue that it was the *cause* of the problem, but this guy never mentioned it.

I fixed the phone number, and sent it back to the support team, with a note electing the guy the "Bonehead user of the week". (*)

(*) OkOkOk... There's probably a very good chance what actually happened in that the guy's kid, seeing his father dial in every night, figured that's how you dial into BBS's also, and changed the phone number sometime when he was home alone with the laptop. When it crashed, he didn't want to admit he touched the laptop, let alone broke it; so he just put it away, and didn't tell anyone.

Share

answered Dec 12, 2008 at 15:18

community wiki
James Curran

6
votes

It was during my diploma thesis. I was writing a program to simulate the effect of high intensity laser on a helium atom using FORTRAN.

One test run worked like this:

- Calculate the intial quantum state using program 1, about 2 hours.
- run the main simulation on the data from the first step, for the most simple cases about 20 to 50 hours.
- then analyse the output with a third program in order to get meaningful values like energy, tork, momentum

These should be constant in total, but they weren't. They did all kinds of weird things.

After debugging for two weeks I went berserk on the logging and logged every variable in every step of the simulation including the constants.

That way I found out that I wrote over an end of an array, which **changed a constant**!

A friend said he once changed the literal 2 with such a mistake.

Share

answered Jun 6, 2009 at 20:44

community wiki
Jens Schauder

5
votes

A deadlock in my first multi-threaded program!

It was very tough to find it because it happened in a thread pool. Occasionally a thread in the pool would deadlock but the others would still work. Since the size of the pool was much greater than needed it took a week or two to notice the first symptom: application completely hung.

Share

answered Oct 4, 2008 at 13:28

community wiki
Rômulo Ceccon

---

5 votes

I have spent hours to days debugging a number of things that ended up being fixable with literally just a couple characters.

Some various examples:

1. ffmpeg has this nasty habit of producing a warning about "brainfart cropping" (referring to a case where in-stream cropping values are >= 16) when the crop values in the stream were actually perfectly valid. I fixed it by adding three characters: "h->".
2. x264 had a bug where in extremely rare cases (one in a million frames) with certain options it would produce a random block of completely the wrong color. I fixed the bug by adding the letter "O" in two places in the code. Turned out I had mispelled the name of a #define in an earlier commit.

Share

answered Oct 5, 2008 at 8:53

community wiki
Dark Shikari

5
votes

My first "real" job was for a company that wrote client-server sales-force automation software. Our customers ran the client app on their (15-pound) laptops, and at the end of the day they dialed up to our unix servers to synchronize with the Mother database. After a series of complaints, we found that an astronomical number of calls were dropping at the very beginning, during authentication.

After weeks of debugging, we discovered that the authentication *always* failed if the incoming call was answered by a getty process on the server whose Process ID contained an even number followed immediately by a 9. Turns out the authentication was a homebrew scheme that depended on an 8-character string representation of the PID; a bug caused an offending PID to crash the getty, which respawned with a new PID. The second or third call usually found an acceptable PID, and automatic redial made it unnecessary for the customers to intervene, so it wasn't considered a significant problem until the phone bills arrived at the end of the month.

The "fix" (ahem) was to convert the PID to a string representing its value in *octal* rather than decimal, making it impossible to contain a 9 and unnecessary to address the underlying problem.

Share

answered Oct 28, 2008 at 0:51

community wiki
Adam Liss

Ya'know... That application (and your name) sound very familiar..... – James Curran Dec 1, 2008 at 20:31

@James: Yes, they do. I was wondering when you'd make the connection! Hope you're doing well ... still "decorating that line with flowers"? – Adam Liss Dec 2, 2008 at 4:49

@Adam: Absolutely. Also, checking your LinkedIn, it seems we passed again. I was at Ratitan for 6 months 2006-07 (and my sister worked at Cryptek in VA) – James Curran Dec 12, 2008 at 14:40

5
votes

Basically, anything involving threads.

I held a position at a company once in which I had the dubious distinction of being one of the only people comfortable enough with threading to debug nasty issues. The horror. You should have to get some kind of certification before you're allowed to write threaded code.

Share

answered Dec 12, 2008 at 15:58

community wiki
PeterAllenWebb

5
votes

I heard about a classic bug back in high school; a terminal that you could only log into if you sat in the chair in front of it. (It would reject your password if you were standing.)

It reproduced pretty reliably for most people; you could sit in the chair, log in, log out... but if you stand up, you're denied, every time.

Eventually it turned out some jerk had swapped a couple of adjacent keys on the keyboard, E/R and C/V IIRC, and when you sat down, you touch-typed and got in, but when you stood, you had to hunt 'n peck, so you looked at the incorrent labels and failed.

Share

answered May 13, 2009 at 20:24

community wiki
Colin

4
votes

While I don't recall a specific instance, the toughest category are those bugs which only manifest after the system has been running for hours or days, and when it goes down, leaves little or no trace of what caused the crash. What makes them particularly bad is that no matter how well you think you've reasoned out the cause, and applied the appropriate fix to remedy it, you'll have to wait for another few hours or days to get any confidence at all that you've really nailed it.

Share

answered Oct 4, 2008 at 4:08

community wiki
Jason Etheridge

4 And invariably it takes at least 3 iterations of that to actually get the right bug fixed... – tloach Oct 7, 2008 at 19:01

I have a bug that happen randomly, and about once every month. Every time I think I fixed it, it comes to bite me back. That sole bug made me fix already about 50 different bugs, but not this bug. – speeder Sep 29, 2010 at 16:16

---

4 votes

Our network interface, a DMA-capable ATM card, would very occasionally deliver corrupted data in received packets. The AAL5 CRC had checked out as correct when the packet came in off the wire, yet the data DMAd to memory would be incorrect. The TCP checksum would generally catch it, but back in the heady days of ATM people were enthused about running native applications directly on AAL5, dispensing with TCP/IP altogether. We eventually noticed that the corruption only occurred on some models of the vendor's workstation (who shall remain nameless), not others.

By calculating the CRC in the driver software we were able to detect the corrupted packets, at the cost of a huge performance hit. While trying to debug we noticed that if we just stored the packet for a while and went back to look at it later, the data corruption would magically heal itself. The packet contents would be fine, and if the driver calculated the CRC a second time it would check out ok.

We'd found a bug in the data cache of a shipping CPU. The cache in this processor was not coherent with DMA, requiring the software to explicitly flush it at the proper times. The bug was that sometimes the cache didn't actually flush its contents when told to do so.

Share

answered Oct 4, 2008 at 4:52

community wiki
DGentry

1 Which begs the question: What led you to store the packet for a while and CRC it later? – Jim Nelson Dec 11, 2008 at 8:09

1 We put the CRC check in the interrupt handler first, and it would detect corrupted packets. Under load, it would also make the system reset from spending too much time at interrupt level, so we moved it out into an asynchronous kernel thread... and the delay in calculating the CRC made a difference. – DGentry Dec 12, 2008 at 4:18