Heisenbug

From Wikipedia, the free encyclopedia
(Redirected from Unusual software bug)

Heisenbug is a computer programming jargon term for a software bug that seems to disappear or alter its behavior when one attempts to study it.[1] The term is a pun on the name of Werner Heisenberg, the physicist who first asserted the observer effect of quantum mechanics, which states that the act of observing a system inevitably alters its state.

Similar terms, such as **bohrbug**, **mandelbug**,[2][3][4] and **schrödinbug**[5][6] have been occasionally proposed for other kinds of unusual software bugs, sometimes in jest;[7][8] however, unlike "heisenbug", they are not widely known or used.[9]

Contents

Examples

Heisenbugs occur because common attempts to debug a program, such as inserting output statements or running it in a debugger, usually modify the code, change the memory addresses of variables and the timing of its execution.

One common example of a heisenbug is a bug that appears when the program is compiled with an optimizing compiler, but not when the same program is compiled without optimization (as is often done for the purpose of examining it with a debugger). While debugging, values that an optimized program would normally keep in registers are often pushed to main memory. This may affect, for instance, the result of floating-point comparisons, since the value in memory may have smaller range and accuracy than the value in the register. Similarly, Heisenbugs may be caused by side-effects in test expressions used in runtime assertions in languages such as C and C++, where the test expression is not evaluated when assertions are turned off in production code using the NDEBUG macro.

Other common causes of heisenbugs are using the value of a non-initialized variable (which may

change its address and/or initial value during debugging), or following an invalid pointer (which may point to a different place when debugging). Debuggers also commonly provide watches or other user interfaces that cause additional source code (such as property accessors) to be executed stealthily, which can, in turn, change the state of the program.

Time can also be a factor in heisenbugs, particularly with multi-threaded applications. Executing a program under control of a debugger can change the execution timing of the program as compared to normal execution. Time-sensitive bugs such as race conditions may not occur when the program is slowed down by single-stepping source lines in the debugger. This is particularly true when the behavior involves interaction with an entity not under the control of a debugger, such as when debugging network packet processing between two machines and only one is under debugger control.

Heisenbugs can be viewed as instances of the observer effect in information technology. Frustrated programmers may humorously blame a heisenbug on the phase of the moon,[10] or (if it has occurred only once) may explain it away as a soft error due to alpha particles or cosmic rays affecting the hardware.

Related terms

A bohrbug, by opposition, is a "good, solid bug". Like the deterministic Bohr atom model, they do not change their behavior and are relatively easily detected.[11][12]

A mandelbug (named after Benoît Mandelbrot's fractal) is a bug whose causes are so complex it defies repair, or makes its behavior appear chaotic or even non-deterministic.

A schrödinbug (named after Erwin Schrödinger and his thought experiment) is a bug that manifests itself in running software after a programmer notices that the code should never have worked in the first place.

A hindenbug [13] (named after the Hindenburg disaster) is a bug with catastrophic behavior.

History of the term

The term was used in 1985 by Jim Gray, in a paper about software failures[14] (and is sometimes mistakenly attributed to him because of this publication) and also in 1986 by Jonathan Clark and Zhahai Stewart on the mailing list (later Usenet news group) comp.risks.[15]

Bruce Lindsay, a researcher at IBM, affirmed in a 2004 ACM Queue interview that he was present when the Heisenbug was originally defined.[16]

An earlier appearance in ACM publications is from 1983.[17]

Resolution

Heisenbugs are usually resolved through very careful debugging. This works best if one is able to identify the approximate point in code where the bug is occurring. From there, solutions may be sought from inspection of nearby statements or analysis of process dumps.

Another technique is to examine logs, especially those produced by lint and lint-like tools.

For highly persistent heisenbugs, it may be necessary to analyze the entire program as a nondeterministic finite automaton to determine their cause.

See also

- Cargo cult programming
- CHESS—a tool for detecting and reproducing Heisenbugs (Windows)
- Memory debugger
- Jinx—a tool that automatically explores executions likely to expose Heisenbugs

References

1. ^ "The Jargon File: heisenbug" (http://catb.org/jargon/html/H/heisenbug.html).
2. ^ "The Jargon File: Mandelbug" (http://catb.org/jargon/html/M/mandelbug.html). Catb.org. Retrieved 2013-09-05.
3. ^ Raymond, Eric S.; *The New Hacker's Dictionary* (http://books.google.ca/books?id=g80P_4v4QbIC&pg=PA295&dq=mandelbug), 3rd edition, 1996
4. ^ Clarke, Arthur C., *The Ghost from the Grand Banks* (http://www.google.ca/search?tbm=bks&q=%22The+Mandelbug.+Get+Ada+to+explain+it+to+you+someday.%22), Bantam Books, 1990
5. ^ "The Jargon File: Schroedinbug" (http://www.catb.org/jargon/html/S/schroedinbug.html). Catb.org. Retrieved 2013-09-05.
6. ^ Raymond, Eric S.; *The New Hacker's Dictionary* (http://books.google.ca/books?id=g80P_4v4QbIC&pg=PA397&dq=schroedinbug), 3rd edition, 1996
7. ^ The following article investigates the various definitions of bohrbug, mandelbug and heisenbug proposed in the literature, as well as the statements made about the relationships between these fault types: Grottke, Michael; and Trivedi, Kishor S.; *Software Faults, Software Aging and Software Rejuvenation*, Journal of the Reliability Engineering Association of Japan, Vol. 27, No. 7, pp. 425-438, 2005.
8. ^ Grottke, Michael; and Trivedi, Kishor S.; *Fighting Bugs: Remove, Retry, Replicate, and Rejuvenate* (http://wayback.archive.org/web/20100327174716/http://www.computer.org/portal/web/csdl/doi/10.1109/MC.2007.55), IEEE Computer vol. 40, no. 2 (February 2007), pp. 107-109
9. ^ A February 2012 Google Books search returns about 70 hits for "schroedinbug", 100 for "mandelbug", 400 for "bohrbug" or "heisenbug".
10. ^ CATB.org, "phase of the moon" (http://www.catb.org/jargon/html/P/phase-of-the-moon.html)
11. ^ Goshgarian, Gary; *Exploring Language*, HarperCollins College Publishers, 1995

12. ^ "Such transient software failures have been given the whimsical name “Heisenbug” because they disappear when reexamined. By contrast, “Bohrbugs” are good solid bugs." (IEEE Computer Group News, Volume 24, Numbers 7–12, 1991)
13. ^ "Hinden Bug" (http://c2.com/cgi/wiki?HindenBug).
14. ^ Gray, Jim (1985). "Why Do Computers Stop And What Can Be Done About It?" (http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.6561). *Technical Report 85.7*. Tandem Computers.
15. ^ (16 December 1986) RISKS DIGEST 4.30 (http://groups.google.com/group/mod.risks/browse_thread/thread/cc68771824a79d3f) - (23 December 1986) RISKS DIGEST 4.34 (http://groups.google.com/group/mod.risks/browse_thread/thread/83a6ad844eda93e0/5e061e6da0c2dbfc?lnk=st&q=heisenbug&rnum=896#5e061e6da0c2dbfc), moderated by Peter G. Neumann
16. ^ " "A Conversation with Bruce Lindsay", ACM Queue vol. 2, no. 8 - November 2004" (http://queue.acm.org/detail.cfm?id=1036486). Queue.acm.org. Retrieved 2013-09-05.
17. ^ *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on High-Level Debugging, Pacific Grove, California, March 20–23, 1983*, Association for Computing Machinery, 1983, Google Books search (http://www.google.ca/search?tbm=bks&q=an+instance+of+such+a+bug+was+called+a+%22Heisenbug%22+by+one+participant):

> “ This is the Heisenberg Uncertainty Principle as applied to debugging (an instance of such a bug was called a "Heisenbug" by one participant.) ”

Also cited in LeBlanc, Richard J.; Robbins, Arnold D.; *Event-Driven Monitoring of Distributed Programs*, in *Proceedings of the IEEE 5th International Conference on Distributed Computing Systems (ICDCS)*, IEEE Computer Society, Computer Society Press, 1985, pp. 515-522 Google Books search (http://www.google.ca/search?tbm=bks&q=This+the+Heisenberg+Uncertainty+Principle+as+applied+to+Debugging%2C+sometimes+called+the+%22Heisenbug%22+Principle+%5BACM83%5D):

> “ This the Heisenberg Uncertainty Principle as applied to Debugging, sometimes called the "Heisenbug" Principle [ACM83]. ”

External links

- The Heisenberg Debugging Technology (http://sourceware.org/gdb/talks/esc-west-1999/)
- A Story About Magic (http://ftp.sunet.se/jargon/html/magic-story.html)

Categories: Software bugs | Debugging | Software testing
