Famous Bugs courtesy of Dave Curry (ecn.davy@purdue).

Originally From John Shore (shore@nrl-css)

Some time ago, I sent out a request for documented reports on "famous bugs", promising to summarize the results for all who contributed. This is the promised summary.

I judge the effort as a failure. People encounter bugs and fix bugs and talk about bugs, but they rarely document bugs. Those responding to my request were well meaning and helpful, but I was left feeling a little like my hair was being done by a gossip-laden ex-hacker.

I am, of course, no different. I didn't have documentation for most of the famous bugs I knew about, and I don't have sufficient time, energy, and interest to follow all the leads. I should have known better.

One strong conclusion that I've reached is that many computer system horror stories become known as bugs when in fact they are not -- they're system problems that result from hardware failures, operator mistakes, and the like. Let me mention a few examples. In his book Software Reliablility, Glenford Myers mentioned a number of classical software errors. For example, he mentioned that a software error in the onboard computer of the Apollo 8 spacecraft erased part of the computer's memory. I happen to have a copy of a memo by Margaret Hamilton that summarized the conclusions of a detailed study of every Apollo bug (see below), so I looked under Apollo 8. The best I could find was this: "16. Illegal P01. P01 was selected during ..dcourse, illegally by the astronaut. This action destroyed W-matrix (several erasables in AGC)." The software did erase the memory, but it did so because the astronaut did something illegal, and not because the programmer goofed. This example is characteristic of the Apollo errors (see below), most of which show the need for better exception handling as part of the software specification. But weak specifications are not the same thing as bugs.

Here's another example, also from Apollo. It starts with a note to me from Kaeler.pa at Xerox (via Butler Lampson):

> I heard about the MIT summer student at NASA whose Apollo program filled up memory (with logging information) and flashed the red abort light a few seconds before the first moon landing. The student was in the control room, and after a little thought, said, "Go ahead, I know what it is, I will take responsibility". Luckily, he was right. He was awarded all sorts of medals for being willing to take the responsibility.
>
> You should get this story from the horse's mouth before distributing it. I heard it from Neil Jacobstein (212-454-0212, in new york). He heard it from his advisor at the Johnson Space Center in Houston a few years ago. It might be interesting to trace this back to someone who really knows about it.

I called Jacobstein, and after some discussion he decided that the "bug" was probably the famous "1201 descent alarm" that I mention below. Again, this was caused by an astronaut error (a radar was turned on

when it should have been off).

Lot's of people mentioned to me various NORAD "bugs" that caused alerts. I got a copy of a Senate Armed Services Committee Report of 9 October, 1980, "Recent False Alerts From the Nation's Missile Attack Warning System." It deals primarily with the June 1980 alerts, but it contains the following summary:

> Oct. 3, 1979 -- An SLBM radar (Mt. Hebro) picked up a low orbit rocket body that was close to decay and generated a false launch and impact report.
>
> November 9, 1979 -- False indications of a mass raid caused by inadvertent introduction of simulated data into the NORAD Computer System.
>
> March 15, 1980 -- Four SS-N-6 SLBMs were launched from the Kuril Islands as part of Soviet troop training. One of the lauches generated an unusual threat fan.
>
> June 3, 1980 -- False indications caused by a bad chip in a communications processor computer

According to Borning@Washington (who by the way is studying computer problems in missile warning systems), the cause of the Nov. 1979 problem was as follows:

> To test the warning system, false attack data was intermixed with data from actual satellite observations, put on tape, and run through the system. On November 9, the test tape of this sort was accidentally left mounted on a secondary backup computer. This machine was left connected to the system in use. When the primary computer failed, a backup computer was activated, which also failed. Then the secondary computer came into play, causing the alert.

All of these missile alerts were caused by real flying objects, hardware failures, or human error. I'm not saying that bugs didn't cause any missile alerts, just that the ones that are reputed to have been caused by bugs in fact were not.

Perhaps computer software -- as opposed to the combination of hardware, software, and people -- is more reliable than folklore has it. I would be interested in hearing your comments on this proposition.

Despite the foregoing problems, the assembly of responses makes interesting reading. In the following, I'll mention a few well-documented bugs and then append various extracts from what I received. Thanks to all who responded. In most cases, I eliminated duplicates. Special thanks to Peter Neumann (SRI), who seems to be keeping better track of these problems than anyone else. Many of these don't qualify as bugs by anyone's definition, but they're interesting stories so I've included some of them anyway.

Space-Shuttle Bug

This may be the most famous of all -- the one that delayed at the last minute the launch of the first space shuttle. The cause was was a bug that interfered with the communication between concurrent processes -- an area of progamming that is among the least well in hand. John R. Garman wrote about the problem in detail in ACM SIGSOFT Software Engineering News (SEN), vol. 6, No. 5, pp 3-10.

The First Bug

Worth mentioning for the trivia folks, it was a moth that was beaten to death by a relay in the Mark II. It was discovered by the Mark II staff, which included Grace Hopper. (Reports on this are a bit confusing. Many attribute the bug to her; in her own published account she refers to "we". I called and asked her. She said the machine operator actually pulled the moth out, and that it was found by the combined efforts of the staff.) Lots of people mentioned this bug to me; my favorite report attributed the bug to

> "some little old lady who works for the Navy. She's the one who came up with Cobol, I think."

In fact, this is one of the better-documented bugs. You can even see its picture in the Anals of the History of Computing (vol. 3, July 1981, page 285). It's ironic that "modern" bugs have practically nothing in common with the first one (an exception being Dijkstra's well known remark about testing).

ARPANET Gridlock

In October 1980 the net was unusable for a period of several hours. It turned out that the routing processes in all the IMPs were consuming practically all resources as the result of processing three inconsistent routing updates. It turned out that the inconsistency arose from dropped bits in a single IMP. Whether you choose to call this a bug or not, clearly it demonstrated a design failure. The details are reported well by Eric Rosen in SEN, January 1981.

APOLLO flight experiences

When Margaret Hamilton was working at the Charles Stark Draper Laboratory in the early 1970s, she documented and analyzed in some detail the various "software anomalies" that occurred during several APOLLO flights. Apparently she did this at the request of Marty Shooman. I don't think that she ever published the results, but some years back she gave us a copy of "Shuttle Management Note #14" (23 October 1972), which summarized her analysis. It makes interesting reading.

One of her strongest conclusions was that 73% of the problems were

caused by "real-time human error". Translated roughly into 1983 computer-speak, this means that the APOLLO software wasn't user friendly. I guess they didn't have icons or something.) Apparently, there was much debate about this during the design, but the software types were told that astronauts have the right stuff or something so there was no need to make the software robust.

One example is quite widely known, as it occured during the APOLLO 11 landing on the moon. In what was referred to as "1201-1202 Descent Alarms", the software kept restarting as the result of overloading. Turned out the radar switch was in the wrong position and used up 13% more computer time than had been anticipated.

Hamilton states that "pure software errors" were not a problem on APOLLO flights. I guess she means that the software met its specifications, which is quite an accomplishment. But the specifications apparently did not say much about error detection and recovery. Hamilton states that "all potentially catastrophic problems would have been prevented by a better and/or known philosophy of providing error detection and recovery via some mechanism."

Nuclear Reactor Design Program

I don't know when the bug was first introduced, but it transpired in 1979. From Jim Horning (Horning.pa@parc-maxc):

belatedly-discovered bug in a stress analysis program (converting a vector into a magnitude by summing components--rather than summing absolute values; module written by a summer student) caused a number of nuclear reactors to be closed down for checks and reinforcement a about three years ago (not long after TMI). This was fairly widely discussed in the press at the time, but I never did see how the lawsuits came out (if, indeed, they have been completed).

>From br10@cmu-10a came the following newswire stories:

a023 0026 16 Mar 79
PM-Topic, Advisory,
Managing Editors:
Wire Editors:
It all started in a tiny part of a computer program used by an engineering firm designing nuclear reactors. It ended with the shutdown of five nuclear power plants at a time when President Carter is pushing oil conservation and the world oil market is in turmoil.
The computer miscalculated some safety precautions required by law. The power from the closed plants now will have to be replaced by electicity generated with oil or coal. This may cost utility customers money and throw a curve at Carter's conservation program.
In Today's Topic: The Little Computer and the Big Problem, AP writer Evans Witt traces this glitch in the system, from the obscure computer to its possible effect on the nation's energy problems.
The story, illustrated by Laserphoto NY7, is upcoming next.
The AP

ap-ny-03-16 0328EST
****

a024 0044 16 Mar 79
PM-Topic-Glitch, Bjt,950
TODAY'S TOPIC: The . yyter and the Big Problem
Laserphoto NY7
By EVANS WITT
Associated Press Writer

WASHINGTON (AP) - Something just didn't add up.

And the result is: five nuclear power plants are shut down; millions of Americans may pay higher utility bills; and a sizable blow may have been struck to President Carter's efforts to reduce the use of imported oil and to control inflation.

The immediate source of all this is part of the federal bureaucracy - the Nuclear Regulatory Commission which ordered the shutdowns.

But in one sense, the ultimate culprit was ''Shock II,'' a tiny part of a computer program used by a private firm to design the power plants' reactors.

Shock II was wrong and that means parts of the five reactors might not survive a massive earthquake. Shock II was the weak link that could have allowed the chain to snap.

In between Shock II and the shutdowns were a public utility, a private engineering firm and the NRC staff. It was really the judgments of the dozens of scientists and engineers, not elected or appointed officials, that led to the shutdowns.

Perhaps as a result, the decision's impact on the nation's energy situation was not even considered until the very last moment - when the commission itself was faced with the final decision.

And at that point, the NRC said, it had no choice. It said the law was clear: serious questions about the reactors had been raised and the reactors had to be turned off until answers were found.

The specific questions are arcane engineering issues, but the explanation is straightfoward: Will some of the systems designed to protect the reactor survive an earthquake - or will they fail, and possibly allow radioactive death to spew into the air?

The regulations say the reactors must be able to withstand a quake equal to the strongest ever recorded in their area. The regulations don't allow any consideration of the likelihood of a major quake. All four states where the reactors are located - New York, Pennsylvania, Maine and Virginia - have had minor quakes in this decade and damaging quakes at least once in this century.

The only way to test them - short of having a massive earthquake - is to test a model of the reactor. The ''model'' is actually a set of mathematical formulas in a computer that reflect how the reactor and its parts will behave in a quake.

The model used for the five reactors came from Stone and Webster, the large Boston engineering and architectural firm that designed the plants. The Stone and Webster model indicated how strong and well supported pipes had to be and how strong valves had to be.

The problem apparently cropped up after Stone and Webster suggested within the last few months more pipe supports in the secondary cooling system of the reactor at Shippingport, Pa., operated by Duquesne Light Co. in Pittsburgh.

But why were the supports needed? ''This was not clear to us, looking at the calculations done by the models,'' said Gilbert W.

Moore, Duquesne's general superintendent of power stations.

So Dusquesne - and Stone and Webster - sent the computer models through their paces again, having them calculate and recalculate what would happen to the pipes in an earthquake.

''We came out with some numbers which were not in the range we would like,'' Moore said.

That made the problem clear - the model now said the pipes might break in an earthquake. The previous analysis indicated an adequate safety margin in the pipes, and Stone and Webster's explanation was: ''One subroutine may not give uniformly conservative results.''

The problem was in a ''subroutine,'' a small part of the computer model, called ''Shock II,'' said Victor Stello, director of NRC's division of reactor operations.

''The facts were that the computer code they were using was in error,'' said Stello. ''Some of the computer runs were showing things are okay. In some cases, the piping systems were not okay.

''We didn't know the magnitude of the error or how many plants might be affected,'' he said.

It was on March 1 that Duquesne told the NRC of the problem by telephone and asked for a meeting to discuss it. The same day, Energy Secretary James R. Schlesinger was telling Congress that unleaded gas might cost $1 a gallon within a year and service stations might be ordered shut down on Sundays because of oil shortages.

The meeting took place on Thursday, March 8, in Washington with NRC staff, Stone and Webster engineers and Duquesne Light people on hand.

Through the weekend, Stello said, engineers from NRC, Duquesne and Stone and Webster worked at the private firm's Boston office, analyzing the severity of the problem.

''By the middle of Sunday (March 10) we begin to get a pretty good idea of what it meant for the systems,'' Stello said. ''Monday, we got the latest information from our people at the Stone and Webster offices. It became clear that there would be a number of the safety systems that would have stresses in excess of allowable limits. The magnitude of the excess was considerable.''

Tuesday, members of the NRC were briefed by their staff of engineers and scientists. They asked for an analysis of the economic impact of the decision, and then ordered the plants closed within 48 hours.

And the five reactors shut down: Duquesne Light Co.'s Beaver Valley plant at Shippingport, Pa.; Maine Yankee in Wiscasset, Maine; the Power Authority of New York's James Fitzpatrick plant at Scriba, N.Y.; and two Virginia and Electric Power Co. reactors at Surry, Va.

It may take months to finish the analysis of the potential problems and even longer to make changes to take care of the situation.

Until the reactors start generating again, the utilities will have to turn to plants using oil or coal. This may cost more, and that cost may be borne by the millions of utility customers.

To replace the power from these nuclear plants could require 100,000 barrels of oil a day or more. And this at a time when President Carter has promised to cut U.S. oil consumption by 5 percent - about 1 million barrels a day - and when the world's oil markets are in turmoil because of recent upheavals in Iran.

Summary of various problems from NEUMANN@SRI-AI

Review of Computer Problems -- Catastrophes and Otherwise

As a warmup for an appearance on a SOFTFAIR panel on computers and human safety (28 July 1983, Crystal City, VA), and for a new editorial on the need for high-quality systems, I decided to look back over previous issues of the ACM SIGSOFT SOFTWARE ENGINEERING NOTES [SEN] and itemize some of the most interesting computer problems recorded. The list of what I found, plus a few others from the top of the head, may be of interest to many of you. Except for the Garman and Rosen articles, most of the references to SEN [given in the form (SEN Vol No)] are to my editorials.

SYSTEM --
SF Bay Area Rapid Transit (BART) disaster [Oct 72]
Three Mile Island (SEN 4 2)
SAC: 50 false alerts in 1979 (SEN 5 3);
  simulated attack triggered a live scramble [9 Nov 79] (SEN 5 3);
  WWMCCS false alarms triggered scrambles [3-6 Jun 80] (SEN 5 3)
Microwave therapy killed arthritic patient by racing pacemaker (SEN 5 1)
Credit/debit card copying despite encryption (Metro, BART, etc.)
Remote (portable) phones (lots of free calls)

SOFTWARE --
First Space Shuttle launch: backup computer synchronization (SEN 6 5 [Garman])
Second Space Shuttle operational simulation: tight loop on cancellation
  of early abort required manual intervention (SEN 7 1)
F16 simulation: plane flipped over crossing equator (SEN 5 2)
Mariner 18: abort due to missing NOT (SEN 5 2)
F18: crash due to missing exception condition (SEN 6 2)
El Dorado: brake computer bug causing recall (SEN 4 4)
Nuclear reactor design: bug in Shock II model/program (SEN 4 2)
Various system intrusions ...

HARDWARE/SOFTWARE --
ARPAnet: collapse [27 Oct 1980] (SEN 6 5 [Rosen], 6 1)
FAA Air Traffic Control: many outages (e.g., SEN 5 3)
SF Muni Metro: Ghost Train (SEN 8 3)

COMPUTER AS CATALYST --
Air New Zealand: crash; pilots not told of new course data (SEN 6 3 & 6 5)
Human frailties:
  Embezzlements, e.g., Muhammed Ali swindle [$23.2 Million],
    Security Pacific [$10.2 Million],
    City National, Beverly Hills CA [$1.1 Million, 23 Mar 1979]
  Wizards altering software or
critical data (various cases)

SEE ALSO A COLLECTION OF COMPUTER ANECDOTES SUBMITTED FOR the 7th SOSP
  (SEN 5 1 and SEN 7 1) for some of your favorite operating system
  and other problems...

[Muni Metro Ghosts]

The San Francisco Muni Metro under Market Street has been plagued with problems since its inauguration. From a software engineering point of

view, the most interesting is the Ghost Train problem, in which the signalling system insisted that there was a train outside the Embarcadero Station that was blocking a switch. Although in reality there was obviously no such train, operations had to be carried on manually, resulting in increasing delays and finally passengers were advised to stay above ground. This situtation lasted for almost two hours during morning rush hour on 23 May 1983, at which point the nonexistent train vanished as mysteriously as it had appeared in the first place. (The usual collection of mechanical problems also has arisen, including brakes locking, sundry coupling problems, and sticky switches. There is also one particular switch that chronically causes troubles, and it unfortunately is a weakest-link single point of failure that prevents crossover at the end of the line.)

Problems mentioned in the book Software Reliability, Glen Myers

Myers mentions a variety of problems. One famous one (lot's of people seem to have heard about it) is the behavior of an early version of the ballistic missile early warning system in identifying the rising moon as an incoming missile. Myers points out that, by many definitions, this isn't a software error -- a problem I discussed a bit at the beginning of this message.

Other problems mentioned by Myers include various Apollo errors I've already mentioned, a 1963 NORAD exercise that was incapacitated because "a software error casued the incorrect routing of radar information", the loss of the first American Venus probe (mentioned below in more detail). Mentioned with citations were an Air Force command system that was averaging one software failure per day after 12 years in operation, deaths due to errors in medical software, and a crash-causing error in an aircraft design program. I was not able to follow up on any of the citations.

The rest of this message contains excerpts from things I received from all over, in most cases presented without comments.

---

faulk@nrl-css (who got it elsewhere)

Today, I heard good story that is a perfect example of the problems that can arise whe the assumptions that one module makes about another are not properly documented. (The story is from a system engineer here whose father got it from the Smithsonian Space people.)

Aparently, the Jupiter(?) probe to mars could have programs beamed to it which it would load in internal memory. The system engineers used this property to make mission changes and/or corrections. After the probe had been on Mars for a while, memory started getting tight. One of the engineers had the realization that they no longer needed the module that controled the landing so the space could be used for something else. The

probe was sent a new program that overwrote the landing module. As soon as this was accomplished, all contact was lost to the probe.

Looking back into the code to find what had gone wrong, the programmers discovered that because the landing module had to have information on celestial navigation, some or all of the celestial navigation functions were inclded in the landing module. Unfortunately, the antenna pointing module also required celestial navigation information to keep the antenna pointed at earth. To do this, it use the navigation functions in the landing module. Overlaying the module has left the antenna pointing in some unknown direction and all contact with the craft has been lost forever.

Fortunately, all of the mission requirements had been fulfilled so it was no great loss. It can live on as a great example of bad design.

---

mo@LBL-CSAM

The folklore tells of a bug discovered during the fateful flight of Apollo 13. It seems that the orbital mechanics trajectory calculation program had a path which had never been excercised because of the smooth, gentle orbital changes characteristic of a nominal Apollo flight. However, when the flight dynamics team was trying ways to get them home with the aid of much more creative maneuvers, the program promplty crashed with a dump (running on IBM equipment, I believe). The story goes that the fix was simple - something on the order of a missing decimal, or a zero-oh reversal, (Divide by zero!!!!!) but there was much consternation and tearing of hair when this critical program bought the farm in the heat of the moment.

This was related to me by an ex-NASA employee, but I have heard it through other paths too. I guess the NASA flight investigation summary would be one place to try and verify the details.

---

jr@bbncd

One cute one was when the Multics swapper-out process swapped out the swapper-in process. (recall that all of the Multics OS was swappable)

---

dan@BBN-UNIX

Here in Massachusetts we've recently begun testing cars for emissions. All car inspections are carried out at gas stations, which in order to participate in the program had to buy a spiffy new emissions analyzer which not only told you what your emissions were, but passed judgement on you as well, and kept a recorrd on mag tape which was sent to the Registry of Motor Vehicles so that they could monitor compliance.

Well, on June 1 the owners of the cheaper ($8K) of the two acceptable analyzers discovered that their machines could not be used; they didn't like the month of June! The company which built them, Hamilton, had to apply a quick fix which

told the machines that it was actually December (!?). Lots of people were inconvenienced.

Unfortunately all I know about this at the moment is what the Boston Globe had to say, so I don't know what the actual problem was. The article said that the quick cure involved replacing the "June" chip with the "December" chip; I don't know what that means, if anything. Electronic News or Computerworld ought to have more accurate information.

Don't forget about the rocket launch early in the space program which had to be aborted because the Fortran program controlling it believed that the number of seconds in a day was 86400 (rather than the sidereal time figure).

The recent issue of Science News with the cover story on when computers make mistakes mentioned a story about a graduate student who almost didn't get his thesis due to inaccuracies in the university computer's floating point software. Not really earthshaking, except to him, I suppose.

STERNLIGHT@USC-ECL

I don't have the data, but there were at least two "almost launches" of missiles. The rising moon was only one. You might try contacting Gus Weiss at the National Security Council--he will be able to tell you quite a bit. Mention my name if you like.

[I called Weiss, who didn't have much to say. He kept repeating that the problems were fixed--js]

---

mark@umcp-cs

The following is probably not famous except with me, but deserves to be.

True story:

Once upon a time I managed to write a check so that my bank balance went exactly to zero. This was not so unusual an occurance, as my checking account had an automatic loan feature in case of overdraft, and I used this feature occasionally. Negative and positive balances were therefore well known in this account. Not so zero balances.

Soon after writing this check I attempted to withdraw some funds using my money machine card. Unsuccessful. I attempted to DEPOSIT money via the machine. Unsuccessful. I talked to a person: they had no record of my account ever having existed.

After several trips to the bank, each time up one more level in the management hierarchy, I, the bank managers and me, discovered the following: The bank's computer had been programmed so that the way to delete an account was to set the balance to zero. When I wrote my fatal zeroing check the computer promptly forgot all about me. Only my passion for paper records, and the bank's paper redundancy, enabled the true story to emerge and my account to be restored.

Interestingly, no funds were ever in danger, since the account was closed with NO MONEY in it. Nonetheless, the convenience was considerable. Once the situation became straightened out I immediately transferred my account to another bank, writing a letter to the first bank explaining my reasons for doing so.

---

craig@umcp-cs

The most famous bug I've ever heard of was in the program which caluclated the orbit for an early Mariner flight to Venus. Someone changed a + to a - in a Fortran program, and the spacecraft went so wildly off course that it had to be destroyed.

---

fred@umcp-cs

> Some examples of bugs I've heard about but for which I don't have documentation: (a) bug forced a Mercury astronaut to fly a manual re-entry; . . .

There was something about this on the Unix-Wizards mailing list a while back. The way I understand it, a programmer forgot that the duration of the Mercury Capsule's orbit had been calculated in siderial time, and left out the appropriate conversion to take into account the rotation of the Earth beneath the capsule. By the end of the mission the Earth had moved several hundred miles from where it ``should'' have been according to the program in question. Sorry I can't give you any definite references to this.

KROVETZ@NLM-MCS

I've heard of two bugs that I think are relatively famous:

1. A bug in a FORTRAN program that controlled one of the inner planet fly-bys (I think it was a fly-by of Mercury). The bug was caused because the programmer inadvertently said DO 10 I=1.5 instead of DO 10 I=1,5. FORTRAN interprets the former as "assign a value of 1.5 to the variable DO10I". I heard that as a result the fly-by went off course and never did the fly-by! A good case for using variable declarations.

2. I'm not sure where this error cropped up, but one of the earlier versions of FORTRAN a programmer passed a number as an actual argument (e.g. CALL MYPROC(2)) and within the procedure changed the formal argument. Since FORTRAN passes arguments by reference this had the result of changing the constant "2" to something else! Later versions of FORTRAN included a check for changing an argument when the actual is an expression.

---

uvicctr!dparnas
From jhart Thu Jun 9 13:30:48 1983
To: parnas

San Fransisco Bay Area Rapid Transit, reported in Spectrum about two years ago. "Ghost trains", trains switched to nonexistent lines, and best of all the rainy day syndrome.

---

uvicctr!uw-beaver!allegra!watmath!watarts!geo

One of my pals told me this story. One morning, when they booted, years ago, the operators on the Math faculty's time-sharing system set the date at December 7th, 1941 (ie Pearl Harbor). Well the spouse of the director of the MFCF (ie Math Faculty Computing Facility) signed on, was annoyed by this, and changed the date to the actual date. Everyone who was signed on while this was done was charged for thirty-something years of connect time.
I wouldn't know how to document this story.

Oh yes, didn't Donn Parker, self-proclaimed computer sleuth, call the fuss made over UNIX and intelligent terminals some outrageous phrase, like 'The bug of the century'? I am refering to the fuss made over the fact that some of the terminals that berkeley had bought were sufficiently intelligent, that they would do things on the command of the central system. The danger was that if someone was signed on to one of these terminals as root, an interloper could write something to this terminal causing the terminal to silently transmit a string back to UNIX. Potentially, this string could contain a command line giving the interloper permissions to which they were not entitled.

Cordially, Geo Swan, Integrated Studies, University of Waterloo
allegra!watmath!watarts!geo

---

smith@umcp-cs

John, another bug for your file. Unfortunately it is a rumor that I I haven't tried to verify. Recall that FORTRAN was developed on the IBM 704. One of the 704's unusual features was that core storage used signed magnitude, the arithmatic unit used 2's complement, and the index regesters used 1's complement. When FORTRAN was implemented on the IBM product that replaced the 704, 7094 etc. series, the 3 way branching if went to the wrong place when testing negative zero. (It branched negative, as opposed to branhing to zero). I heard this rumor from Pat Eberlein (eberlein@buffalo). Supposedly, the bug wasn't fixed (or descovered) for two years.

VES@KESTREL

1. In the mid 70's in a lumber provessing plant in Oregon a program was controlling the cutting of logs into boards and beams. The program included an algorithm for deciding the most efficient way to cut the log (in terms of utilizing most of the wood), but also controlled the speed with which the log was advancing.

Once the speed of a log increased to dangerous levels. All personnel

was scattered and chased out of the building, the log jumped off the track, fortunately there were no casualties. This was caused by a software bug. reference to the event would be the former director of the Computer Center at Oregon State University (prior to 1976), who at the time I heard the story (Spring 1977) was President of the company which developed the software.

2. Abother rather amusing incident dates back to the mid 60's. It was not caused by a software bug but is indicative of the vulnerability of software systems particularly in those early days. It involved the Denver office of Arizona airlines. Their reservation system was periodically getting garbage input. Software experts were dispatched but failed to identify the cause. Finally the software manager of the company which developed the system went to study the problem on the spot.

After spending a week at the site he managed to identify a pattern in the generation of garbage input: it was happening only during the shifts of a paticular operator and only when coffie was served to her. Shortly afterwards the cause was pinpointed. The operator was a voluminous lady with a large belly. The coffie pot was placed behind the terminal and when she would reach for it her belly would rest on the keyboard. Unfortunately, I don't have more exact references to that event.

RWK at SCRC-TENEX

There's the famous phase-of-the-moon bug which struck (I believe) erry Sussman and Guy Steele, then both of MIT. It turned out to e due to code which wrote a comment into a file of LISP forms, that included the phase of the moon as part of the text. At certain times of the month, it would fail, due to the comment line being longer than the "page width"; they had failed to turn off automatic newlines that were being generated by Maclisp when the page width was exceeded. Thus the last part of the line would be broken onto a new line, not proceeded with a ";" (the comment character). When reading the file back in, an error would result.

---

gwyn@brl-vld

An early U.S. Venus probe (Mariner?) missed its target immensely due to a Fortran coding error of the following type:

```
DO 10 I=1.100
```

Which should have been

```
DO 10 I=1,100
```

The first form is completely legal; it default-allocates a REAL variable DO10I and assigns 1.1 to it!

ɔrning.pa@PARC-MAXC

I have heard from various sources (but never seen in print) the story

that the problem with the wings of the Lockheed Electras (that caused several fatal crashes) slipped past the stress analysis program because an undetected overflow. This one would probably be next to impossible to document.

One of my favorite bugs isn't all that famous, but is instructive. In about 1961, one of my classmates (Milton Barber) discovered that the standard integer binary-to-decimal routine provided by Bendix for the G-15D computer wasn't always exact, due to accumulated error from short multiplications. This only affected about one number in 26,000, but integer output OUGHT to be exact. The trick was to fix the problem without using any additional drum locations or drum revoltions. This occupied him for some time, but he finally accomplished it. His new routine was strictly smaller and faster. But was it accurate? Milton convinced himself by numerical analysis that it would provide the correct answer for any number of up to seven digits (one word of BCD). Just to be safe, he decided to test it exhaustively. So he wrote a loop that counted in binary and in BCD, converted the binary to BCD, and compared the results. On the G-15D this ran at something like 10 numbers per second. For several weeks, Milton took any otherwise idle time on the college machine, until his loop had gone from 0 to 10**7-1 without failure. The he proudly submitted his routine to Bendix, which duly distributed it to all users. Soon thereafter, he got a telephone call: "Are you aware that your binary to decimal conversion routine drops the sign on negative numbers?" This is the most exhaustive program test that I've ever seen, yet the program failed on half its range!

..rning.pa

[Excerpts from a trip report by Dr. T. Anderson of the University of Newcastle
upon Tyne.]

The purpose of my trip was to attend a subworking group meeting on the production of reliable software, sponsored by NASA, chaired by John Knight
(University of Virginia), and organized and hosted by the Research Triangle
Institute [Nov. 3-4, 1981]. Essentially, NASA would like to know how on earth
software can be produced which will conform to the FAA reliability standards of
10^-9 failures/hour. Sadly, no one knew.

FRANK DONAGHE (IBM FEDERAL SYSTEMS DIVISION): PRODUCING RELIABLE SOFTWARE FOR THE SPACE SHUTTLE

Software for the Space Shuttle consists of about 1/2 million lines of code,
produced by a team which at its largest had about 400 members. Costs were high
at about $400 per line. . . . Between the first and second flight 80% of the
mdoules were changed, such that about 20% of the code was replaced. Three

weeks prior to the second flight a bug in the flight software was detected which
ed up the four primary computers in a tight (two instruction) loop. .

Laws@SRI-AI

Don't forget the bug that sank the Sheffield in the Argentine war. The shipboard computer had been programmed to ignore Exocet missiles as "friendly." I might be able to dig up an IEEE Spectrum reference, but it is not a particularly good source to cite. The bug has been widely reported in the news media, and I assume that Time and Newsweek must have mentioned it.

I'll try to find the reference, but I must issue a disclaimer: some of my SRI associates who monitor such things more closely than I (but still without inside information) are very suspicious of the "bug" explanation. "Computer error" is a very easy way to take the heat off, and to cover what may have been a tactical error (e.g., turning off the radar to permit some other communication device to function) or a more serious flaw in the ship's defensive capability.

PARK@SRI-AI

From International Defense Review and New Scientist after the lklands war ...

The radar system on the Sheffield that didn't report the incoming Exocet missile because it wasn't on the list of missiles that it expected a Russian ship to use.

The missiles fired over the heads of British troops on the Falklands beaches at the Argentinians, that could have gone off if they had detected enough metal below them (probably not really software).

The missile that was guided by a person watching a tv picture of the missile from a shipboard camera. A flare on the tail of the missile intended to make the missile more visible to the camera tended to obscure the target. A hasty software mod made the missile fly 20 feet higher (lower?) so that the operator could see the target.

A more general consideration is that in the event of an electromagnetic pulse's deactivating large numbers of electronic systems, one would prefer that systems like missiles in the air fail safe.

Laws@SRI-AI

I have scanned Spectrum's letters column since the original Exocet mention in Oct. '82, but have not found any mention
the bug. Perhaps I read the item on the AP newswire or saw a newspaper column posted on a bulletin board here at SRI. Possibly Garvey@SRI-AI could give you a reference. Sorry.

---

rvey@SRI-AI
Subject: Re: Falkland Islands Bug

The original source (as far as I know) was an article in New Scientist (a British magazine) on 10 Feb 83. It suggested that the Exocet was detected by the Sheffield's ESM gear, but catalogued as a friendly (!) missile, so no action was taken. I have two, strictly personal (i.e., totally unsubstantiated by any facts or information whatsoever) thoughts about this:

1) I suspect the article is a bit of disinformation to cover up other failings in the overall system; from bits and pieces and rumors, I would give top billing to the possibility of poor spectrum management as the culprit;

2) I wouldn't care if the missile had the Union Jack on the nose and were humming Hail Brittania, if it were headed for me, I would classify it as hostile!

PALMER@SCRC-TENEX

I was working for a major large computer manufacturer {not the one that employees me today}. One of the projects I handled was an RPG compiler that was targeted to supporting customers that were used to IBM System III systems. There had been complaints about speed problems from the field on RPG programs that used a table lookup instruction. The computer supporting the compiler had excellent micorocde features: we decided to take advantage of the feature by microcoding the capability into the basic machine.

The feature was pretty obvious: it would search for things in ordered or unordered tables and do various things depending on whether the key was in the table.

We made what we considered to be the "obvious" optimization in the case of ordered tables - we performed a binary search. Nothing could be faster given the way the tables were organized. We wrote the code, tested it on our own test cases and some field examples and got performance improvements exceeding sales requirements. It was an important fix...

Unfortunatly, it was wrong. It isn't clear what "it" was in this case - Us or IBM. People loaded their tables in the machine with each run of an RPG program and they often wouldn't bother to keep their ordered tables ordered. IBM didn't care - it ignored the specification {most of the time}. Our code would break when people gave us bad data in ways that IBM's wouldn't. We had to fix ours.

Olmstread.PA@PARC-MAXC

can't supply any documentation, but I was told when I was in school with Ron Lachman (you might want to check with him at LAI -- laidback!ron, I think) that IBM had a bug in its program IEFBR14.

This program's sole job was to return (via BR 14, a branch through register 14, the return register); it was used by JCL (shudder!) procedures which allocated file space and needed a program, any program, to run. It was a one-line program with a bug: it failed to clear the return code register (R 15, I think). I submit you won't find any programs with a higher bugs-per-instruction percentage.

---

hoey@NRL-AIC

I got this at MIT....

From: ihnp4!zehntel!zinfandel!berry@ucb-vax

In the April 1980 issue of ACM SIGSOFT Software Engineering Notes, editor Peter G. Neumann (NEUMANN@SRI-KL at that time) relays information that Earl Boebert got from Mark Groves (OSD R&E) regarding bugs in the software of the F-16 fighter. Apparently a problem in the navigation software inverted the aircraft whenever it corssed the equator. Luckily it was caught early in simulation testing and promptly fixed.
In the July issue, J.N. Frisina at Singer-Kearfott wrote to Mr. Neumann, "concerned that readers might have mistakenly believed there was a bug in the flight software, which was of course not the case." [At least they fixed THAT one. Wasn't it Hoare who said that acceptance testing is just an unsuccessful attempt to find bugs?] Mr. Frisina wrote:

"In the current search for reliable software, the F16 Navigation software is an example of the high degree of reliability and quality that can be obtained with the application of proper design verification and testing methodologies. All primary misison functions were software correct."

In the April '81 Issue it is revealed that the F18 range of control travel limits imposed by the F18 software are based on assumptions about the inability of the aircraft to get into certain attitudes. Well, some of these 'forbidden' attitudes are in fact attainable. Apparently so much effort had gone design and testing of the software that it is now preferable to modify the aircraft to fit the software, rather than vice-versa!

Cantone@nrl-aic

I've heard from Prof. Martin Davis, a logician at NYU, that Turing's Ph.D. thesis was just filled with bugs. His thesis was a theoretical description of his Turing machine that included sample computer programs for it. It was these programs that were filled with bugs. Without computers there was no way to check them.
(Those programs could have worked with only minor fixes).

[NOTE: I called Davis, who gave as a reference a paper by Emile Post on recursive unsolvability that appeared in 1947-8 in the Journal of Symbolic Logic -- js]

David.Smith@CMU-CS-IUS

In simulation tests between the first and second Shuttle flights, a bug was found in the onboard computer software, which could have resulted in

the premature jettison of ONE of the SRB's. That would have been the world's most expensive pinwheel!

i read this in Aviation Week, but that leaves a lot of issues to scan.

---

butkiewi@nrl-css

In response to your request for info on bugs, here's one.
We work with some collection system software that was initially deployed in 1974. Part of the system calculated the times of upcoming events. In 1976, on February 29th, some parts of the system thought it was a different julian day and it basically broke the whole system. Several subroutines needed leap year fixes. One software engineer was called in from home and worked all night on that one. How many sound Software Engineering Principles were violated?

Stachour.CSCswtec@HI-MULTICS

while i cannot cite any published documentation, and this could hardly qualify as a famous bug, a mail-system I once worked on (which ran with priviledge to write into any-one mailboxes) was discovered to have an incorrect check for message-length froma message coming from a file. The result was that a 'specially prepared msg' could arrange to overlay he test of the mail-system, and especilly restore the system change-access' program on top of the mail-system, which then gave the caller power to change access controls on any file of the system.
This was for a university-built mail-system for a HOneywell GCOS3 ciria 1976.

Sibert@MIT-MULTICS

I imagine you've heard about this already, and, if not, I can't provide any documentation, but anyway: it is said that the first Mariner space probe, Mariner 1, ended up in the Atlantic instead of around Venus because someone omitted a comma in a guidance program.

Kyle.wbst@PARC-MAXC

TRW made a satellite in the late '50's or early '60's with the feature that it could power down into a stand-by mode to conserve electrical consumption. On the first pass over the cape (after successful orbital check out of all systems), the ground crew transmitted the command to power down. On the next pass, they transmitted the command to power up and nothing happened because the software/hardware system on board the satellite shut EVERYTHING down (including the ground command radio receiver).

Hoffman.es@PARC-MAXC

>From the AP story carried on Page 1 of today's Los Angeles Times:

Jet Engine Failure Tied to Computer: It's Too Efficient

The efficiency of a computer aboard a United Airlines 767 jet may have led to the failure of both of the plane's engines, forcing the aircraft into a four-minute powerless glide on its approach to Denver, federal officials said Tuesday.

. . .

[The National Transportation Safety Board's] investigation has disclosed that the overheating problem stemmed from the accumulation of ice on the engines.

. . .

[I]t is believed that the ice built up because the onboard computer had the aircraft operating so efficiently during the gradual descent that the engines were not running fast enough to keep the ice from forming.

. . .

The incident raised questions among aviation safety experts about the operation of the highly computerized new generation of jetliners that are extremely fuel-efficient because of their design and computerized systems.

"The question is at what point should you averride the computer," one source close to the inquiry said.

. . .

[T]he engines normally would have been running fast enough to keep the ice from forming. In the case of Flight 310, investigators believe, the computer slowed the engine to a rate that conserved the maximum amount of fuel but was too slow to prevent icing.

key question, one source said, is whether the computer-controlled descent might have kept the flight crew from recognizing the potential icing problem. Airline pilots for some time have complained that the highly computerized cockpits on the new jets -- such as the 767, Boeing's 757 and the Airbus 310 -- may make pilots less attentive.

. . .

Kaehler.pa@parc-maxc
Info-Kermit@COLUMBIA-20

On Wednesday, August 24, at 11:53:51-EDT, KERMIT-20 stopped working on many TOPS-20 systems. The symptom was that after a certain number of seconds (KERMIT-20's timeout interval), the retry count would start climbing rapidly, and then the transfer would hang. The problem turns out to be a "time bomb" in TOPS-20. Under certain conditions (i.e. on certain dates, provided the system has been up more than a certain number of hours), the timer interrupt facility stops working properly. If KERMIT-20 has stopped working on your system, just reload the system and the problem will go away. Meanwhile, the systems people at Columbia have developed a fix for the offending code in the monitor and have distributed it to the TOPS-20 managers on the ARPAnet.

The problem is also apparent in any other TOPS-20 facility that uses timers, such as the Exec autologout code.

The time bomb next goes off on October 27, 1985, at 19:34:06-EST.

Craig.Everhart@CMU-CS-A has put together a long file of stories that were gathered as the result of a request for interesting, humorous, and socially relevant system problems. They make fun reading, but most of them aren't really bugs, let alone well-docmented ones, so I decided not to include them. The file, however, can be obtained from Craig.

(end-of-message)

-Bob (Krovetz@NLM-MCS)