

b gg Degubbing

I guess I've solved (or tried to solve) more than my share of bizarre problems over the years. It might be useful to review a few more memorable debugging yarns. To see if they can't lead us to some useful guidelines...

The LAN of the Nineties

Um, that's the *eighteen* nineties. Otherwise known as the *Gamewell Fire Alarm Telegraph*. Those little red boxes once found on city street corners everywhere.

Inside each box was an amazingly ingenious fist-sized spring wound clockwork. *Elegant Simplicity* at its very finest. Each node box had its own id url (spelling out their location code), offered collision detection and avoidance (patiently waiting until another box was through), network reconfiguration (you could bypass chosen boxes for service) and even did adaptive routing (sending through ground if the series line burned up). Why, these even had the world's first Baud Rate. Yup. *One Baud*.

I was a Parker fireman at the time, and their tape punch would start rarely and randomly dumping indecipherable messages on the apparatus floor. Strangely, there was not a peep out of our wake-the-dead *Santa Fe* air horns on the same system. Suggesting a short rather than an open.

Month by month, things slowly got worse. Eventually we concluded this was wind related. Naturally, the wind in Parker only blows strongly at 2 AM once every few weeks. Tracing every inch of the 37,000 foot long system (dumbly) seemed like something I'd want to avoid.

Instead, I used the box programming pins to first bypass half of the system. Of course, pushing any pin created an immediate dead calm. Weather control at its finest.

Eventually, I narrowed it down to the line between two boxes. One that went twelve feet up in the air, down the street, an alley, then another street to the next box.

How could you have a random dead short twelve feet in the air? Careful inspection quickly found the culprit. The power drop to a home included a steel cable that lightly touched the alarm wire. Wind movement scraped through, causing the maddingly infuriating intermittent.

A piece of garden hose gave an immediate repair, and the power company reworked things a few days later.

Synchronized Four-Wheeling

At least for me, those VW Syncro 4WD vans were the finest vehicles anytime ever. These could literally go anywhere (they *never* spun their wheels), were economical, perfect for sleeping in, were Bowseretta friendly, and so huge you

never had to empty them. Your stuff always fit.

Just about twenty nanoseconds after the last honest VW dealer in Arizona went belly up, my Syncro started bucking and heaving about an hour into any trip. Travel to the far away remaining dealers racked up thousands of dollars in charges that did nothing. Except for trashing gas springs by an incompetent mechanic. And my finding out that their service director did not even know that the Syncro *had* a computer, let alone knowing where it was.

I quickly found out about the superb *Bentley Manuals* and used these to learn the VW electronics. Thankfully, you could easily overhaul a Syncro engine while it was still running down the thruway (ask my daughter about her "backseat driver" story the time the accelerator cable broke on a long, long trip) Anyway, I found I could push pins through various wires and monitor them with a VOM.

Which verified a probable computer or sensor problem.

I decided to lash up a 12 volt inverter so I could use an oscilloscope to find out which signals were going wrong how. Just as Bee was pulling out the driveway, I hit the computer with my fist. The engine came to a dead stop.

A plain old intermittent! The usual *Radio Shack* tuner cleaner on all the contacts did little good, so I decided to resolder every joint in the computer. The problem quickly became obvious: Two power resistors with steel leads were not wet properly during production.

An instant fix on a problem that, to this day VW has not got the faintest clue over.

Demolishing eexec

When *Adobe Systems* first came out with *PostScript* for their laser printers, they stupidly went to exceptional pains to prevent you from accessing and intelligently using all of the *font paths* you needed for 3-D perspective lettering or for similar *Nonlinear Transformations*.

They did this by introducing a spectacularly useless and mind-numbingly annoying *eexec* encryption scheme.

Which Adobe felt was "absolutely uncrackable".

Since it carried its own keys and made a 1:1 mapping, it was hard to see how eexec could really be very secure at all. I quickly found a legal and klutzy workaround that let you approximate font paths by using narrow clipped character slices. Which was slow and imprecise.

It didn't take others long to find the "double clip" bug in early PostScript laser printers. When you clipped within a clip, certain machines forgot to do the *invalidaccess* block on returned font paths.

Because Adobe themselves didn't want to be annoyed by eexec, they built a bypass trapdoor into their code. Which was fast discovered. Word thus got out on the **1183615869 internaldict begin** master key.

Was eexec all that secure? Not in the least. Their highly touted concept had a fundamental and profound flaw in it that any patient seventh grader could easily find.

It seems that PostScript is a general purpose computing language. When the printer was set up for two way comm, **error messages were returned!**

All you had to do was **intentionally** add an extra byte into an eexec string. The returned error message gave a lot of previously decrypted eexec stuff piled up on the stack, followed by a misspelled operator.

It was exactly the same as a safe audibly telling a burglar "Try three more clicks to the left". Simply by progressively introducing "extra character" errors on purpose every few dozen or so eexec characters, you could easily extract all of the font paths and any other "protected" information.

Adobe eventually realized the error of their ways and published full and open eexec info in their **Black Book**. At the same time, the **pathforall** lockout was removed from firmware. PostScript font paths are now accessible.

The Eternally Leaking Hot Tub

I've long been a hot tub enthusiast. Having an older deck unit with external equipment. Over the years, I made a few **upgrades**. Finding out that a timer, manual temp control, and an antithermosiphon valve greatly reduced operating costs. (HINT: Put the one-way valve on the high side or -DUH- your pump will not prime!)

The tub gradually started losing water. At one inch per day, it finally got bad enough that something had to be done. Calculation showed this to be a **plonk ... plonk ... plonk** leak rate rather than something obvious.

The leak stopped when the tub was half empty.

I first touched up some apparently minor cracks with a polyester repair kit, then replaced the drain valve. Neither helped. I next started snooping through the narrow cracks in the deck. First with my fiberscope and then a moisture meter probes. Ultimately, a long thin brass tube brought up mud in exactly the worst place.

Removing a slump block from the adjacent building wall turned into a fiasco as the wrong one was removed. Finally, some boards were split out of the deck. Sure enough, the jet PVC plumbing was wet. But digging out the plumbing made it dry again.

The problem ultimately turned out to be a pinhole in a jet seal gasket. Which was well above the minimum level.

Um, it seems that my hot tub problems are neverending. I hired a local firm to refinish the tub a beautiful blue. But it was the wrong (polyurethane rather than acrylic) plastic, was hygroscopic, and turned a milky white in the presence of warm water. What should have been a two day job now has taken months. But "real soon now"...

The "Tearing Method"

Many summers ago, I worked on an Arizona **Fire Lookout Tower** and had more than enough extra time to carefully brute force hand disassemble an **Adams Adventure**. Using nothing but a pocket card. To solve a situation I could not get past playing fair by the rules. Which led to my **Tearing**

Method of converting object code back into source code.

The tearing method works by **deducing the structure** of a computer program or whatever. And then letting the structure reveal what is really coming down. By using page highlighter color coding for a "divide and conquer" analysis. It first appeared in my **Enhancing your Apple II** and led to both my **Apple Assembly Cookbook** and **Applewriter Cookbooks**. Although it is best for fixed position mixed opcode and data programs, it remains quite useful for **PIC Micros** and elsewhere.

With the tearing method, you first disassembly list the entire program and then separate the "live" code from data areas. Usually the live code will be rational things done in rational ways, and the data (when wrongly interpreted as working opcodes) will produce gibberish.

The Achilles' heel of working code is the **RTS** or return from subroutine. You color these green, breaking the code down into its individual low level modules. Follow this up with orange **JSR** subroutine calls. Then use other colors, possibly blue for branches, red for direct or indirect jumps, yellow for housekeeping.

The most popular low level modules are then related against known tasks and locations. The **Avalanche Effect** then is used to ripple back through the rest of the code.

Over on the data side, tests are applied to find out what data does which. Again reducing the program down to its primitive elements. Using patterns, experience, tests, and the obvious "fill it with water and see where it leaks". For instance, ASCII text will often have lots of hex **\$20** values, along with **\$13** carriage returns or **\$10** linefeeds. You can very quickly learn to "sight read" hex ASCII messages. Many other data structures similarly become obvious. Such as those used for memory location lists, graphic sprites, baud rate tables, or math tables.

The tearing method is amazingly flexible. I've taught it both as CC Adult Education and in grade school computer camps. And, with a few mods, continue to use it to this day in my **Catlog Format** tutorials and **Consulting Work**.

Whoops, There it Went

An **InfoPack** client recently asked me to find those hidden **Data Formats** for the **Radio Shack** wireless humidity and temperature transmitters. Which consisted of a brief data burst every few minutes. Knowing the format could be handy for computer logging or diverting the units into new uses. Such as my **Isopods**

As often happens, I had just **sold** my last storage scope, so I first used the latent persistence of an ordinary scope in a totally dark room. Quickly discovering that a forced reset could send data every three seconds.

And that we were apparently dealing with a Differential Manchester code. The supplemental Radio Shack service manuals were most useful. You often have to scream and shout to get these at certain stores.

I then went an "ancient history" route and then wrote the raw data into an Apple IIe computer by way of its game paddle port. Which quickly and hassle free gave me their raw data stash. A few more bytes of machine language code let me decode their weather info. Which was surprisingly complex and subtle, owing to the elaborate error detection and anti-collision scheme they used.

Further details appear **here**.

"Lady, you picked it"

Sometimes, satisfying your customer has very little to do with a technically correct fix. Getting the people involved to accept your work as a positive experience can often get tricky. But years and years ago, a tv repairman we'll call Owen taught me an important lesson.

In those days, tv service firms made house calls. Sets were heavy and didn't take lightly to being hauled around. Even moving a color set was fraught with peril. But more often than not, replacing a "burned out" tube having an open filament was all you needed for a fix.

And Owen did. But the lady of the house was obviously problem-oriented. She wanted Owen to guarantee that her tv was going to work years into the future. Exasperated, Owen thought for a while and then reached around the set back and slightly misadjusted the then-common vertical hold control. The tv started slowly scrolling frames. Owen then asked her to pick the picture she wanted.

It was a rough decision, and there were several near misses, but she eventually chose the very best one. All by herself. Which Owen locked in and then left.

Lessons Learned

Let us try to gather together several of my hard earned first principles of debugging...

- **The technical solution and the people solution can often be wildly and totally different.**
- **Make intermittents happen! They often cannot be reliably fixed otherwise.**
- **Split the system in half. Then split again. Find out where the problem is **not**. Leave no place to hide.**
- **The onerous task you are studiously avoiding will probably end up the fastest, cheapest, and most reliable way to solve the problem.**
- **What are intentionally introduced errors trying to reveal to you?**
- **To make a micrometer adjustment, hit something with a micrometer.**
- **Divide and conquer. Big lumps first.**
- **Believe what your measurements are trying to tell you! No matter how strange. Nor what they do to your current mindset.**
- **Fast and crude preliminary tests can often end up as highly effective time saving filters.**
- **Always have full schematics and documentation on hand. You can later resell them on eBay.**
- **Never try to get by with less in the way of proper instruments than you really need. If it takes a logic analyzer or an oscilloscope, get and use one.**
- **Take a hike.**
- **Usenet newsgroups and web newsletters can be exceptionally valuable info and help resources. But never trust a sole source.**
- **A library or web hour equals a lab month.**









- **Assume all professional experts are incompetent. This eliminates rude surprises later on.**
 - **Try to make the problem worse.**
 - **The more secure the system, the easier a patient seventh grader can and will trash it.**
 - **Older, simpler, and more brute force methods can sometimes save the day.**
 - **Small input changes can reveal data structures. Fill it with water and see where it leaks.**
 - **Try to be as noninvasive as possible. Always check the most probable causes first.**
 - **Deduce the form and structure and many details will take care of themselves.**
 - **Talk the problem out with someone. Or your dog.**
- and finally, of course...
- **Iffen it ain't broke, don't fix it.**

Microcomputer pioneer and guru Don Lancaster is the author of 35 books and countless articles. Don maintains a US technical helpline you will find at (928) 428-4073, besides offering his own **books**, reprints and **consulting services**.

Don also offers surplus bargains found on **eBay** and on his **Bargain Pages**.

Don is also the webmaster of **www.tinaja.com** You can also reach Don at Synergetics, Box 809, Thatcher, AZ 85552. Or you can use email via **don@tinaja.com**

PLEASE CLICK HERE TO...

-  **Go to the [main library](#)**
-  **Start your [tech venture](#)**
-  **Sponsor a display [banner](#)**
-  **Find [research solutions](#)**
-  **Send Don Lancaster [email](#)**
-  **Pick up surplus [bargains](#)**
-  **Find out what a [tinaja](#) is**
-  **View recommended [books](#)**