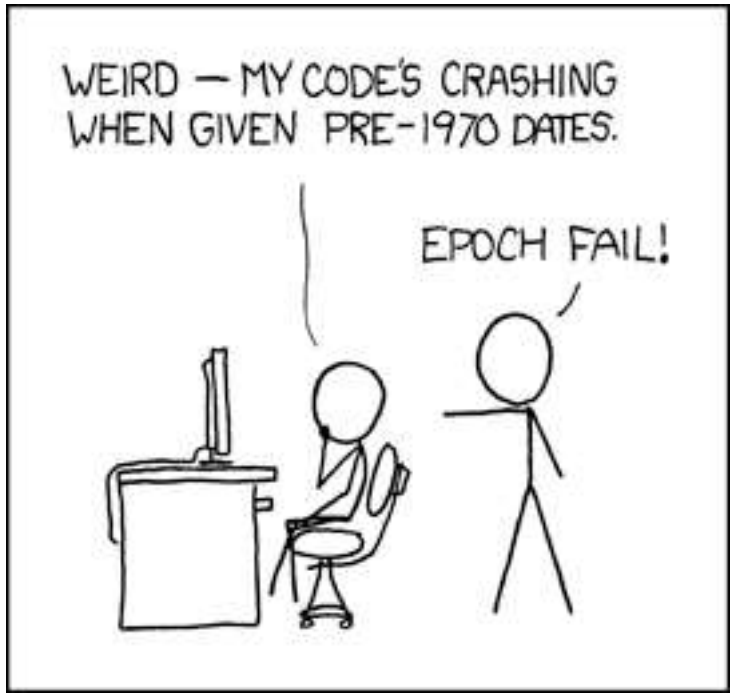
Finding and solving problems in software



Debugging

What is debugging?

"Bug" and "debugging" are attributed to the discovery of a moth found in a Mark II computer at Harvard University.

Debugging is not troubleshooting

Debugging Rules...

1. Understand the system

2. Make it fail

3. Quit thinking and look

4. Divide and Conquer

5. Change one thing at a time

6. Keep an audit trail

7. Check the plug

8. Get a fresh view

9. If you didn't fix it, it aint fixed

Debugging - A Simple Method

a) Start

|

b) Construct a Hypothesis

|

c) Construct an Experiment

|

d) Hypothesis Proved? - goto b)

|

Need more evidence? - goto c)

|

Stop

1. Understanding the System

Know the fundamentals of the system

Understand the frameworks involved

Read the API's and communication interfaces

Know your tools

Understand what each module of the system does

2. Make it Fail

Start at the beginning

Automate the failure

No detail is insignificant

Intermittent failures?

Isolate the failure

Build debugging tools and test harnesses

3. Quit thinking and look

Gather the low level details and "see" the failure

Understand the context of the bug

Design instrumentation in:

- Runtime statistics
- Status messages
- Debug logging - be descriptive!!!
- Deployment time environment checks

Hypothesise and test

4. Divide and Conquer

Narrow the search down, software AND hardware. Successive approximation.

Start at the bad end (e.g. the error log) and work your way up the chain looking at the various points on the way

Fix the bugs you know about - fixing one could fix the other

Fix the noise first, but don't go nuts

5. Change one thing at a time

Compare with a good one !

Use a rifle not a shotgun

What's changed recently, it used to work!

Separate environment and product

6. Keep an audit trail

Keep a day book of what you did

During debugging write down...

1. What you did...
2. In what order...
3. And what happened!

Be specific and consistent

Correlate symptoms with others

Version control and change logs are great for determining an audit trail of what's changed

7. Check the plug

Question your assumptions

8. Get a Fresh View

Get another viewpoint on the solution

Talk to somebody about the problem

Ask an expert:

- Email the author
- Check the forums
- Subscribe to the mailing list

Make sure you report symptoms not theories

9. If you didn't fix it, it aint fixed!

The most important lesson by far...

- Fix the **root cause**
- Make sure it can't possibly happen again
- Fix the process at fault

"We didn't see an issue until we deployed into the production environment"

= Deploy early and deploy often

Debugging Methods and Tools

Debugging Methods

Print/Trace Debugging - watching the flow of execution after adding print statements to the original code

Remote Debugging - connect to a remote system with a debugger, monitor the execution and state

Post Mortem Debugging - analysis of a memory dump from a program that has crashed

Profiling - not debugging per se, but a useful technique for analysing a running program to look for issues

Lesser Known (?) Debugging Methods

Rubber Ducking - explaining your problem to someone else. Stating the problem aloud in order to work it out.

Failfirst - Write a unit test that exposes the failure, correct the code and make it pass

Saff Squeeze - Write a system test, inline code, write test, repeat until fixed

http://www.threeriversinstitute.org/HitEmHighHitEmLow.html

Chaos Monkey - AWS Netflix approach - randomly kill and test areas. Fail consistently to test fault tolerant software.

http://www.codinghorror.com/blog/2011/04/working-with-the-chaos-monkey.html

Eclipse Debugging 1

Set breakpoints in the code

Step into, over, return

Watches - display a variable value

Hit counts

Setting an exception breakpoint

Setting a method breakpoint

Eclipse Debugging 2

Class load breakpoint

Using display after a breakpoint has been hit

Attaching a remote debugger via a JDB agent

Bug Taxonomies and Bug Types

There are ways to categorise bugs... lots of taxonomies available from research literature

Some bug types include:

- Bohrbug - bug that manifests itself consistently under a set of conditions
- Heisenbug - bug that alters its' characteristics when studied

Preventing Bugs

Debugging is hard and expensive in terms of fixing the bug, testing it and re-deploying it...

We can try and prevent bugs with:

- Code Review
- Defensive Programming
- Precondition, postcondition and invariant checking
- Assertions

Further Resources

- Debugging - The 9 Indispensable Rules for Finding Even The Most Elusive Software and Hardware Problems - David Agans
- Debug It! : Find, Repair and Prevent Bugs in Your Code (Pragmatic Programmers) - Paul Butcher
- Testing Education - http://www.testingeducation.org/a/bugtax.pdf
- Lars Vogels Debugging Overview (2011) - http://www.vogella.de/articles/EclipseDebugging/artic