Organisation und Architektur von Rechnern

Foundations of Debugging

Christoph Daniel Schulze

Problems With Software

```
// Update frames
for (int i = 0; i < frames.length; i++) {
    Container containers[] =
        frames[i].getContainers();

    // Update containers
    for (int j = 0; j < containers.length; j++) {
        SwingUtilities.updateComponentTreeUI(
            containers[i]);
    }
}
```

Problems With Software

Ariane 5 (Source: Wikipedia)

Problems With Software

USS Yorktown (CG-48) (Source: Wikipedia)

Outline

Bugs. . .

. . . in computer programming

Wikipedia says:

> *A software bug is an error, flaw, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways.*

A Bit of History

Where does the term ‘bug’ come from?

Legend:

- Moth found in relay of Harvard Mark II computer with Admiral Grace Hopper around in 1947.
- She then supposedly said she “found a bug.”

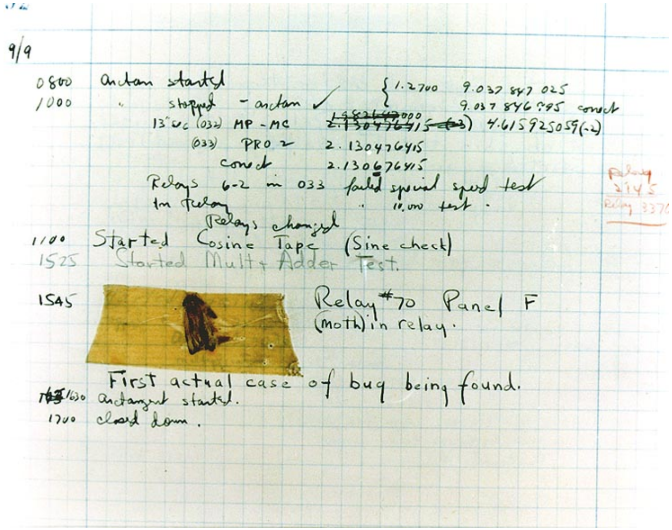
Truth:

- The term ‘bug’ had been in use as early as 1878.
- Grace Hopper popularized the term.

Rear Admiral Grace Hopper (Source: Wikipedia)

The Original Log Book

Moth included



9/9

0800 Arctan started

1000 " stopped - arctan ✓ {1.2700 9.037 847 025

9.037 846 795 correct

13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415~~ ~~(-3)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

correct 2.130676415

Relays 6-2 in 033 failed special speed test

In relay " 11.000 test .

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Multy Adder Test.

1545 Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

~~1630~~ 1630 arctangent started.

1700 closed down.

Relay 2145

Relay 3370

(Source: Wikipedia)

Types of Bugs

Kinds of things that can go wrong

- Arithmetic

  Division by zero, overflow / underflow, loss of precision, . . .
- Logic

  Infinite loops, off-by-one errors, . . .
- Syntax

  Writing `horst = 42` instead of `horst == 42`, . . .
- Resource

  Null pointers, uninitialized variables, access violations, memory leaks, buffer overflows, stack overflows, . . .
- Synchronization

  Deadlocks, race conditions, . . .

Reproducability of Bugs

A measure of frustration

- Deterministically Reproducible

  A fixed sequence of steps reproduces the bug.
- Non-Deterministically Reproducible

  Sometimes it’s there, sometimes it’s not. (often: concurrency problems)
- Heisenbugs

  A bug that is not reproducible anymore once you start debugging.

Ways for Finding Bugs

Manual and automatic

- Software Testing

  Sit down and do stuff to your software. (normal use)
- Aggressive Software Testing

  Sit down and do bad stuff to your software. (explicitly test corner cases)
- Code Analysis

  Have a tool look for common mistakes programmers tend to make.
- Instrumentation

  Analyse your software while it's running.

What to Do Once You’ve Found a Bug

A handy list of steps towards success

1. Reproduce

   Work out a sequence of steps that will cause the bug to appear.
2. Simplify

   Work out a minimal sequence steps that will cause the bug to appear.
3. Deduce

   Form and test hypotheses about the cause of the bug.

   Things that can help:
   - Tracing
   - Debuggers
4. Fix

   Fix the bug! Have a victory beer!

How Tracing Works

…and why it's also called "printf debugging"

Insert `printf(...)` statements into your C code to…

- …ensure correct values of variables.
- …see when (and if) certain pieces of code execute.
- …look for violated assumptions.

Example:

```c
int i;
for (i = 0; i <= 255; i++) {
    // Trace
    printf("Iteration␣%d\n", i);

    // Do stuff
}
```

What Are Debuggers?

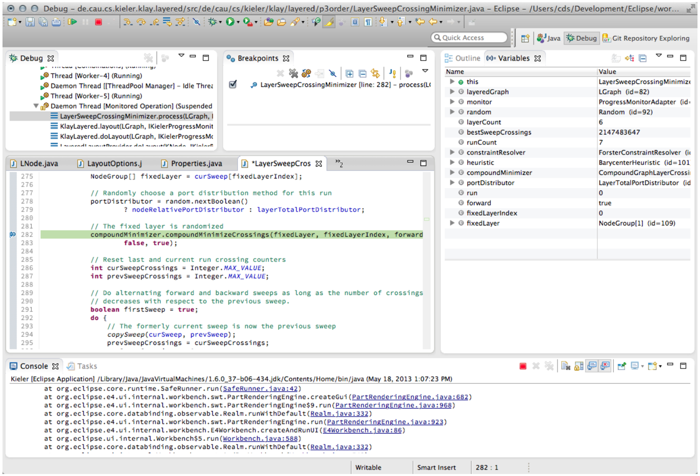
A short overview

Features:

- setting breakpoints
- inspecting and modifying the contents of memory and registers
- single-stepping through your code (forward and backward)
- replacing code as it runs
- remote debugging

Popular Debuggers

Programs you might use in the future



Debug – de.cau.cs.kieler.klay.layered/src/de/cau/cs/kieler/klay/layered/p3order/LayerSweepCrossingMinimizer.java – Eclipse – /Users/cds/Development/Eclipse/wor...

```
Thread [Worker-4] (Running)
Daemon Thread [[ThreadPool Manager] - Idle Threa
Thread [Worker-5] (Running)
Daemon Thread [Monitored Operation] (Suspended
    LayerSweepCrossingMinimizer.process(LGraph, I
    KlayLayered.layout(LGraph, IKielerProgressMonit
    KlayLayered.doLayout(LGraph, IKielerProgressMo
```

☑ LayerSweepCrossingMinimizer [line: 282] - process(L

Name	Value
this	LayerSweepCrossingMinimize
layeredGraph	LGraph (id=82)
monitor	ProgressMonitorAdapter (id=
random	Random (id=92)
layerCount	6
bestSweepCrossings	2147483647
runCount	7
constraintResolver	ForsterConstraintResolver (i
heuristic	BarycenterHeuristic (id=101
compoundMinimizer	CompoundGraphLayerCrossi
portDistributor	LayerTotalPortDistributor (id
run	0
forward	true
fixedLayerIndex	0
fixedLayer	NodeGroup[1] (id=109)

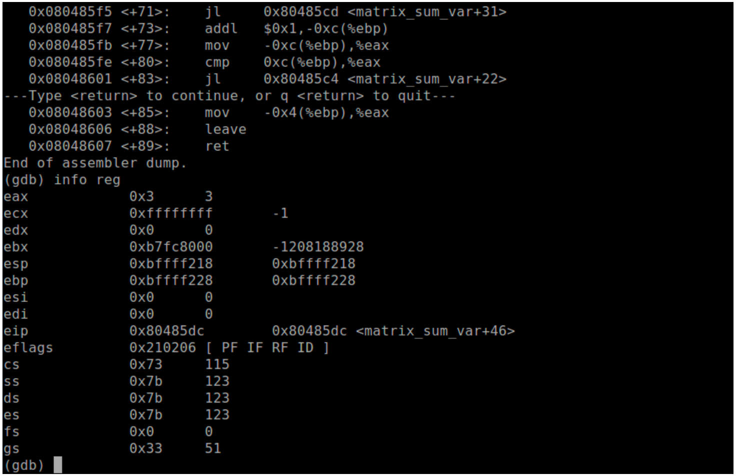
```java
        NodeGroup[] fixedLayer = curSweep[fixedLayerIndex];

        // Randomly choose a port distribution method for this run
        portDistributor = random.nextBoolean()
                ? nodeRelativePortDistributor : layerTotalPortDistributor;

        // The fixed layer is randomized
        compoundMinimizer.compoundMinimizeCrossings(fixedLayer, fixedLayerIndex, forward
                false, true);

        // Reset last and current run crossing counters
        int curSweepCrossings = Integer.MAX_VALUE;
        int prevSweepCrossings = Integer.MAX_VALUE;

        // Do alternating forward and backward sweeps as long as the number of crossings
        // decreases with respect to the previous sweep.
        boolean firstSweep = true;
        do {
            // The formerly current sweep is now the previous sweep
            copySweep(curSweep, prevSweep);
            prevSweepCrossings = curSweepCrossings;
```

```
Kieler [Eclipse Application] /Library/Java/JavaVirtualMachines/1.6.0_37-b06-434.jdk/Contents/Home/bin/java (May 18, 2013 1:07:23 PM)
        at org.eclipse.core.runtime.SafeRunner.run(SafeRunner.java:42)
        at org.eclipse.e4.ui.internal.workbench.swt.PartRenderingEngine.createGui(PartRenderingEngine.java:682)
        at org.eclipse.e4.ui.internal.workbench.swt.PartRenderingEngine$9.run(PartRenderingEngine.java:968)
        at org.eclipse.core.databinding.observable.Realm.runWithDefault(Realm.java:332)
        at org.eclipse.e4.ui.internal.workbench.swt.PartRenderingEngine.run(PartRenderingEngine.java:923)
        at org.eclipse.e4.ui.internal.workbench.E4Workbench.createAndRunUI(E4Workbench.java:86)
        at org.eclipse.ui.internal.Workbench$5.run(Workbench.java:588)
        at org.eclipse.core.databinding.observable.Realm.runWithDefault(Realm.java:332)
```

Graphical: Eclipse Debugger

Popular Debuggers

Programs you might use in the future



```
   0x080485f5 <+71>:    jl     0x80485cd <matrix_sum_var+31>
   0x080485f7 <+73>:    addl   $0x1,-0xc(%ebp)
   0x080485fb <+77>:    mov    -0xc(%ebp),%eax
   0x080485fe <+80>:    cmp    0xc(%ebp),%eax
   0x08048601 <+83>:    jl     0x80485c4 <matrix_sum_var+22>
---Type <return> to continue, or q <return> to quit---
   0x08048603 <+85>:    mov    -0x4(%ebp),%eax
   0x08048606 <+88>:    leave
   0x08048607 <+89>:    ret
End of assembler dump.
(gdb) info reg
eax            0x3      3
ecx            0xffffffff       -1
edx            0x0      0
ebx            0xb7fc8000       -1208188928
esp            0xbffff218       0xbffff218
ebp            0xbffff228       0xbffff228
esi            0x0      0
edi            0x0      0
eip            0x80485dc        0x80485dc <matrix_sum_var+46>
eflags         0x210206 [ PF IF RF ID ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x33     51
(gdb) 
```

Command-line: The GNU Project Debugger (GDB)

Using GDB

A short introduction

Let’s use GDB to debug stuff!

Beyond Debugging

There's more to debuggers than debugging

Fun stuff to do with debuggers:

- wreak havoc by randomly calling functions
- reverse engineer unknown programs