Oops, I did it again - Funny Programming Fails

Lukas Iffländer  Tim Hegemann
December 14, 2016
Funny Programming Fails

static bool isCrazyMurderingRobot = false;

void interact_with_humans (void){
    if(isCrazyMurderingRobot = true)
        kill(humans);
    else
        be_nice_to(humans);
How to accidentally cheat PABS

Toyota - Nothing is Impossible, even Code that Kills

A Story of Knights and Farmers

Funny Takeouts

Broken by Optimization

Santas Sled
How to accidentally cheat PABS
Implement the MergeSort algorithm for arrays of int in Java
@Test
public void testMergeSortSorted() {
    int[] testArray = {1, 2, 3, 4, 5, 6};
    MergeSort.sort(testArray);
    assertArrayEquals("Array not sorted",
                     new int[] {1, 2, 3, 4, 5, 6}, testArray);
}

@Test
public void testMergeSortSortedDesc() {
    int[] testArray = {6, 5, 4, 3, 2, 1};
    MergeSort.sort(testArray);
    assertArrayEquals("Array not sorted",
                     new int[] {1, 2, 3, 4, 5, 6}, testArray);
}
More PABS Tests

```c
// testMergeSort:
int[] testArray = {1, 3, 7, 5, 2, 9};
// [...]

// testMergeSort2:
int[] testArray = {16, 22, 38, 27, 85, 38, 60};
// [...]

// testMergeSort3:
int[] testArray = {7, 75, 24, 20, 12, 54, 19, 42, 73, 81};
// [...]

// testMergeSort4:
int[] testArray = {8, 12, 69, 31, 49, 49, 40, 3, 53, 13, 84, 36, 86, 72, 89, 94, 70};
```
The following code passes ALL six tests:
public static void sort(int[] arr) {
    if (arr.length < 2) return;
    sort(arr, 0, arr.length - 1);
    merge(arr, 0, arr.length / 2, arr.length - 1);
}

static void sort(int[] arr, int start, int end) {
    if (end - start == 1) return;
    int mid = (start + end + 1) / 2;
    sort(arr, start, mid);
    sort(arr, mid, end);
    merge(arr, start, mid, end);
}
```java
static void merge(int[] arr, int start, int mid, int end) {
    while (start < end || mid < end) {
        if (arr[start] <= (arr[mid])) {
            if (start < mid) start++;
            else mid++;
        } else {
            int tmp = arr[mid];
            arr[mid] = arr[start];
            arr[start++] = tmp;
        }
    }
}
```
Try this example:

```java
sort(new int[] {3, 4, 1, 2})
```

Invariant for `merge(...)`:
Both Parts are sorted ⇒ The whole becomes sorted
merge(int[] arr, int start, int mid, int end) {
  true
  while (start < end || mid < end) {
    3 1
    if (arr[start] <= (arr[mid])) {
      if (start < mid) start++;
      else mid++;
    } else { // => swap(0, 2); start++
      int tmp = arr[mid];
      arr[mid] = arr[start];
      arr[start++] = tmp;
    }
  }
}
Debugging

```c
// {1, 4, 3, 2}   1   2   3
merge(int[] arr, int start, int mid, int end) {
//    true
    while (start < end || mid < end) {
//      4   3
        if (arr[start] <= (arr[mid])) {
            if (start < mid) start++;  // => swap(1, 2); start++
                else mid++;
        } else { // => swap(1, 2); start++
            int tmp = arr[mid];
            arr[mid] = arr[start];
            arr[start++] = tmp;
        }
    }
}
```
Debugging

```c
#include <stdlib.h>
#include <stdio.h>

// {1, 3, 4, 2}  2  2  3
void merge(int[] arr, int start, int mid, int end) {
    // true
    while (start < end || mid < end) {
        // 4  4
        if (arr[start] <= arr[mid]) {
            if (start < mid) start++;
        } else {
            int tmp = arr[mid];
            arr[mid] = arr[start];
            start++;
        }
    }
}
```
merge(int[] arr, int start, int mid, int end) {
    true
    while (start < end || mid < end) {
        if (arr[start] <= (arr[mid])) {
            if (start < mid) start++;
            else mid++;
        } else {
            // => swap(2, 3); start++
            int tmp = arr[mid];
            arr[mid] = arr[start];
            arr[start++] = tmp;
        }
    }
}
merge(int[] arr, int start, int mid, int end) {
    while (start < end || mid < end) {
        if (arr[start] <= (arr[mid])) {
            if (start < mid) start++;
            else mid++;
        } else {
            int tmp = arr[mid];
            arr[mid] = arr[start];
            arr[start++] = tmp;
        }
    }
}
• This `merge(...)` method is totally crap!
• Result of `sort({3, 4, 1, 2})` is `{2, 1, 3, 4}`
• Six JUnit tests failed to detect this!
• One of them testing a 17 elements array
Toyota - Nothing is Impossible, even Code that Kills
Unintended Acceleration
Unintended Acceleration

- Toyota cars suddenly accelerate at full power
- Breaking does NOT stop the acceleration
- Only way to stop is handbrake
Breaking distance up to 100 meters!!!

81 deaths so far!
Unintended Acceleration

• Toyota cars suddenly accelerate at full power
• Breaking does NOT stop the acceleration
• Only way to stop is handbrake
  Breaking distance up to 100 meters!!!
• 81 deaths so far!
First Investigation

- Acceleration code investigated by NASA
  - Did not find a “smoking gun”
  - But
    - tight timeline
    - limited information / access (trade secrets)
    - no exoneration of the system

Statement of U.S. Transportation Secretary:

“We enlisted the best and brightest engineers to study Toyota’s electronic systems, and the verdict is in. There is no electronic-based cause for unintended high-speed acceleration in Toyotas.”

Lesson:

Politicians do not know jack shit about software.
First Investigation

- Acceleration code investigated by NASA
  - Did not find a “smoking gun”
  - But
    - tight timeline
    - limited information / access (trade secrets)
    - no exoneration of the system

- Statement of U.S. Transportation Secretary:
  
  “We enlisted the best and brightest engineers to study Toyota’s electronic systems, and the verdict is in. There is no electronic-based cause for unintended high-speed acceleration in Toyotas.”
First Investigation

- Acceleration code investigated by NASA
  - Did not find a “smoking gun”
  - But
    - tight timeline
    - limited information / access (trade secrets)
    - no exoneration of the system

- Statement of U.S. Transportation Secretary:
  
  “We enlisted the best and brightest engineers to study Toyota’s electronic systems, and the verdict is in. There is no electronic-based cause for unintended high-speed acceleration in Toyotas.”

- Lesson: Politicians do not know jack shit about software.
Why did NASA not find Anything

- Software in one chip not analyzed at all. Only main CPU software analyzed.
Why did NASA not find Anything

• Software in one chip not analyzed at all. Only main CPU software analyzed.
• Toyota told NASA they had EDAC (Error Detection and Correction)
Why did NASA not find Anything

- Software in one chip not analyzed at all. Only main CPU software analyzed.
- Toyota told NASA they had EDAC (Error Detection and Correction)
- **But:** There was no EDAC for the RAM
Code “Architecture”

256'600 Non-Commented Lines C Source
39'000 Non-Commented Lines C Headers (Main CPU only)
??? Proprietary Monitor Chip Software
256'600 Non-Commented Lines C Source
39'000 Non-Commented Lines C Headers (Main CPU only)
?? Proprietary Monitor Chip Software

Code only for acceleration!
Software Testing @ Toyota

Testing only at vehicle level.

No

• Unit Testing
• Integration testing
Vehicle Testing not Enough

• Vehicle level testing useful and important
  • Unexpected component interactions
  • Environment influences in real-world application

• Complete testing at vehicle level unpractical
  • Too many combinations of possible conditions, timings
  • Too many possible sources for failures
    • Two faults can counter each other
    • Source of defects hard to locate
Toyota Coding Rules

- 11 of 35 rules suggested for road vehicles found in coding rules
- Rules last updated 1998
- Those weren’t followed:
  - 105 of 343 switch keywords without default
- 14 of 35 rules violated, 7’134 violations
  - Macros
  - Use of #undef
Static Code Analysis

• Coverity
  97 variables declared but not referenced
  5 include recursion

• Codesonar
  2272 global variable declared with different types
  333 cast alters value
  99 condition contains side-effect
  64 multiple declaration of global variable
  22 uninitialized variables

• Uno
  89 possibly uninitialized variable
  2 array of 16 byte initialized with 17 bytes
Spaghetti Code

- McCabe Cyclomatic Complexity Metric
  - Number of “eyes” in flow control graph
  - Unit tests harder with complex graph
  - Over 50 considered “untestable”

- Toyota Code
  - 67 functions with complexity over 50
  - Throttle angle function: **146**
    - 1300 LOC, no test plan
Global Variables

- Ideal Number: ZERO
- Toyota: 9’273 - 11’528 global variables
  - 6’971 local static sufficient
  - 1’086 file static sufficient
Other Issues

- Poor isolation of task functions
- Many large functions
- Reviews informal and only on some modules
- No configuration management
- No bug tracking system
- No formal specification
- Write code you can be confident of it being safe
- You should be able to sleep with the knowledge of software being used in production.
Personal Story
Personal Story
A Story of Knights and Farmers
Singlethreaded Algorithm
Multithreaded Algorithm
The graphic rendered by the multithreaded algorithm is corrupt:

- Some pixels have a different color than they should
- Some pixels have no color at all

⇒ We need synchronization :(
private final Semaphore rendezvous;
private final Queue<Knight> knights;
// Implementation: ConcurrentLinkedQueue

public void runMT(int nThreads) {
    ExecutorService pool =
        Executors.newFixedThreadPool(nThreads);
    do {
        // do some preparation ... fill knight queue
        for (int i = 0; i < nThreads; i++) {
            pool.submit(this::run);
        }
        rendezvous.acquire(nThreads);
    } while (/*work to do*/);
    pool.shutdown();
}
private void run() {
    while (!knights.isEmpty()) {
        Knight knight = knights.remove();
        while (!knight.isSatisfied()) {
            // do stuff ...
        }
    }
    rendezvous.release();
}
The multithreaded variant of the algorithm works (same output as the singlethreaded one)

It is way faster (factor 2.3 on an Intel Core i3 [2C + HTT])
Synchronized Multithreaded Algorithm

- The multithreaded variant of the algorithm works (same output as the singlethreaded one)
- It is way faster (factor 2.3 on an Intel Core i3 [2C + HTT])
- Rendering a 1080p scene randomly fails...
Synchronized Multithreaded Algorithm

- The multithreaded variant of the algorithm works (same output as the singlethreaded one)
- It is way faster (factor 2.3 on an Intel Core i3 [2C + HTT])
- Rendering a 1080p scene randomly fails...
- Rendering a 4K scene always fails
Synchronized Multithreaded Algorithm

- The multithreaded variant of the algorithm works (same output as the singlethreaded one)
- It is way faster (factor 2.3 on an Intel Core i3 [2C + HTT])
- Rendering a 1080p scene randomly fails...
- Rendering a 4K scene always fails
- DEADLOCK
• Deadlock occurs on heavy load
• The only blocking structure is that semaphore we added
• Debugging prints tell us the semaphore sticks because of too few `release()` calls
• ⇒ some threads never finish
private void run() {
    // ----->
    while (!knights.isEmpty()) {
        Knight knight = knights.remove();
        // <----- IS NOT ATOMIC (but should!)

        while (!knight.isSatisfied()) {
            // do stuff ...
        }
    }
    rendezvous.release();
}
private void run() {
    Knight knight;
    while ((knight = knights.poll()) != null) {
        while (!knight.isSatisfied()) {
            // do stuff ...
        }
    }
    rendezvous.release();
}
Working 8K Example
Conclusions

• Threads can disappear when they throw an uncaught exception or error
• Threads from ThreadPools do not even log something to stdout/stderr when they die
• Check for exceptions on your own
• Use Rust
Funny Takeouts
x = 0;
while x < 5
    x = x + 1;
end
%do something with x ...
try {
  //...
} catch (SecurityException sex) {
  //...
}
```java
public class A {
    protected String foo;
    public void setFoo(String fooVal);
    public String getFoo();
    public void doSomething() {
        ...
        foo = x.munge();
        ...
    }
}

public class B extends A {
    /* redeclared here for clarity */
    protected String foo;
    public void doSomething() {
        ...
        foo = x.munge();
        ...
    }
}
```
int getRandomize(int randMax)
{
    srand ( time(NULL) );
    int randNum; = rand() % randMax + 1;
    return 2;
}
```cpp
int multiplyBy10(int number)
{
    stringstream str;
    str << number << '0';
    str >> number;
    return number;
}
```
void get_tomorrow_date( struct timeval *date )
{
  sleep( 86400 ); // 60 * 60 * 24
  gettimeofday( date, 0 );
}
// Not a joke, I've really seen that
for ($i=0 ; $i<3 ; $i++) {
    switch($i) {
    case 1:
        // do some stuff
        break;
    case 2:
        // do some stuff
        break;
    case 3:
        // do some stuff
        break;
    }
}
Double Kill

```javascript
$('body *:visible').hide().show();
$('body *:not(:visible)').show().hide()
```
Broken by Optimization
There is an ancient legend, every programmer knows, that aggressive compiler optimizations break your code.
This legend is true
The example

```c
#include "stdio.h"

int main() {
    int i, j = 0;
    for (i = 1; i > 0; i += i)
        ++j;
    printf("%d\n", j);
}
```
Try the example

$ gcc example.c
$ ./a.out
31
$
At release...

$ gcc -O3 -Wall example.c
$ ./a.out

$
At release...

$ gcc -O3 -Wall example.c
$ ./a.out
^C
$ □
gcc -O0 -S example.c

#include "stdio.h"

int main() {
    int i, j = 0;
    for (i = 1; i > 0; i += i)
        ++j;
    printf("%d\n", j);
}
#include "stdio.h"

int main() {
    int i, j = 0;
    for (i = 1; i > 0; i += i)
        ++j;
    printf("%d\n", j);
}

main:
.LFB11:
    .cfi_startproc
    .p2align 4,,10
    .p2align 3
.L2:
    jmp .L2
    .cfi_endproc
# [...]

58
Well then?

- OK - indeed –03 is very aggressive
Well then?

- OK - indeed –03 is very aggressive
- Trying –02 ...
Well then?

- OK - indeed –03 is very aggressive
- Trying –02 ...
- Same result (even same assembler code!)
Well then?

- OK - indeed –03 is very aggressive
- Trying –02 ...
- Same result (even same assembler code!)
- Then –01 ?!
Well then?

- OK - indeed –03 is very aggressive
- Trying –02 …
- Same result (even same assembler code!)
- Then –01 ?!
- At least this one works:
#include "stdio.h"

int main() {
  int i, j = 0;
  for (i = 1; i > 0; i += i)
    ++j;
  printf("%d\n", j);
}
What happened?

GCC signed integer overflow optimization
What happened?

GCC signed integer overflow optimization

$ gcc -O3 -fno-strict-overflow example.c produces nearly the same assembler code as $ gcc -O1 example.c
This legend is true
But it’s all your own fault :)

Broken by Optimization
Santas Sled
Now, at the end of this talk, let’s have some look at Santa Claus’ sled management software:

- for every reindeer save their name and guide (the reindeer before them)
- save the christmas present for every reindeer
- list all reindeers with the present they get
public class Reindeer {
    private final String name;
    private Reindeer guide;

    public Reindeer(String name) {...}
    public Reindeer getGuide() {...}
    public void setGuide(Reindeer guide) {...}
    public String getName() {...}

    @Override public boolean equals(Object o) {
        // [...]
        return Objects.equals(name, reindeer.name) &&
               Objects.equals(guide, reindeer.guide);
    }

    @Override public int hashCode() {
        return Objects.hash(name, guide);
    }
}
public class SantasPlan {
    Map<Reindeer, String> presents = new HashMap<>();
    Reindeer leader;

    void prepareForChristmas() {
        Reindeer donner = new Reindeer("Donner");
        leader = donner;
        Reindeer comet = new Reindeer("Comet");
        comet.setGuide(donner);
        Reindeer blixen = new Reindeer("Blixen");
        blixen.setGuide(comet);

        presents.put(donner, "noise cancelling headphones");
        presents.put(comet, "a fitness tracker");
        presents.put(blixen, "new sunglasses");
    }
}
```java
class SantasPlan {

  void foggyChristmasEve() {
    Reindeer rudolph = new Reindeer("Rudolph");
    leader.setGuide(rudolph);
    leader = rudolph;
    presents.put(rudolph, "tissues");
  }

  public static void main(String[] args) {
    SantasPlan plan = new SantasPlan();
    plan.prepareForChristmas();
    plan.foggyChristmasEve();
    for (Reindeer reindeer : plan.presents.keySet()) {
      System.out.println(reindeer.getName() + " gets " + plan.presents.get(reindeer));
    }
  }
}
```
Guess what happens?

1. Everything runs well
2. Rudolph does not show up
3. Just Rudolph is shown nobody else
4. Every reindeer is printet but some loose their presents
1. Everything runs well
   Seriously? We’re talking about FAILS!
2. Rudolph does not show up
3. Just Rudolph is shown nobody else
4. Every reindeer is printet but some loose their presents
Guess what happens?

1. Everything runs well
   Seriously? We’re talking about FAILS!

2. Rudolph does not show up
   Why shouldn’t he?

3. Just Rudolph is shown nobody else

4. Every reindeer is printet but some loose their presents
Guess what happens?

1. Everything runs well
   Seriously? We’re talking about FAILS!

2. Rudolph does not show up
   Why shouldn’t he?

3. Just Rudolph is shown nobody else
   Getting closer...

4. Every reindeer is printet but some loose their presents
1. Everything runs well
   Seriously? We’re talking about FAILS!

2. Rudolph does not show up
   Why shouldn’t he?

3. Just Rudolph is shown nobody else
   Getting closer...

4. Every reindeer is printet but some loose their presents
   You got it!
Output of main

Blixen gets null
Rudolph gets tissues
Donner gets null
Comet gets null
@Override
public int hashCode() {
    return Objects.hash(name, guide);
}

Where is our bug?

- Adding Rudolph as Donner’s guide alters Donners hashcode
- Altering Donner’s hashcode alters Comet’s hashcode …
- The HashMap stores the presents under the old hashcodes
- But looks them up calculating the new ones
Where is our bug?

- Adding Rudolph as Donner’s guide alters Donner's hashcode
- Altering Donner’s hashcode alters Comet’s hashcode ...
- The HashMap stores the presents under the old hashcodes
- But looks them up calculating the new ones
- $\Rightarrow$ FAIL!
Merry Christmas

Murray Christmas