Comp 11 Lectures

Mike Shah

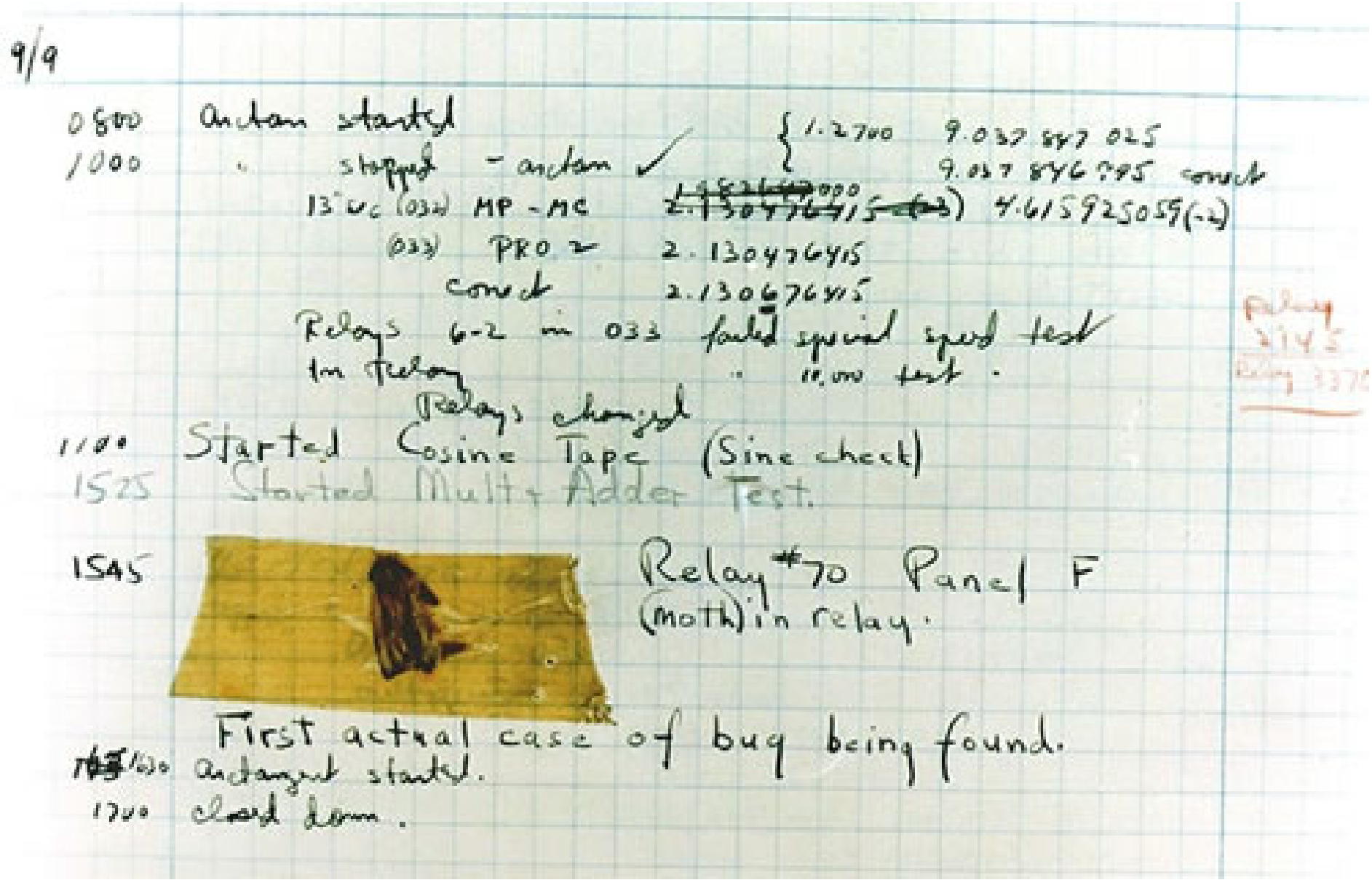
Tufts University

July 26, 2017

Please do not distribute or host these slides without prior permission.

Testing

Comp 11 - Pre-Class warm up



Lecture

Grace Hopper

Figure 2: Grace Hopper was one of the first famous programmers, working on the Harvard Mark 1 machine (Just across the river). She coined the term computer "bug" when she found a machine not running because a moth got stuck in it. Grace also led development on the COBOL programming language, and made many contributions to the field of compilers.

Lecture

Testing Software

Software Correctness

"Testing can show the presence of errors, but not their absence." E. W. Dijkstra

- With this in mind, it is hard to prove that our software is correct.
- For small programs we may be okay, but we need some techniques that will help us as we work on larger software projects.

Famous Software Bugs[1] - Case 1

`http://listverse.com/2012/12/24/10-seriously-epic-computer-software-bugs/`

- Therac-25 Medical Accelerator
- It had two types of radiation therapy.
- One setting would apply a strong dosage, one a lower dosage.
- If the operator changed modes of the devices two quickly, the wrong one would be applied!

[1] The term coming from the moth we previously saw

Famous Software Bugs - Case 2

- North American blackout was due to a generator going out.
- The software bug however, was that the system failed to sound the alarms (it failed quietly)–and the situation could have been mitigated much quicker.

Figure 3: August 14, 2003

Famous Software Bugs - Case 3

- Y2K Bug
- 2 digits were used to represent the year instead of 4, which would break some calculations.
- This was attributed to short-sightedness on programmers. Had they taken this course, they would know data representation matters! :)
- (We are going to have this same problem in Unix systems in 2038 actually based on how 32-bit Unix systems store time)

Figure 4: Y2K was a seemingly simple bug conceptually that required lots of maintenance!

Famous Software Bugs - Case 4

- Kerberos Random Number Generator from 1988-1996
- Unfortunately, this did not generate truly random numbers based on the seed.
- The seed was always the same, so if a computer uses *randomness* for authentication, the system is compromised!

Famous Software Bugs - Case 5

- Intel Pentium floating point divide
- Dividing numbers within a specific range yielded a small error.
- Critical software systems could be compromised! (especially if you divide a lot!)
- This bug is estimated to cost Intel $475 million over the bugs lifetime. Not a small chunk of change!

Tools for testing

- So for us, we need some tools that can help prevent bugs!
- We have actually been exposed to one through some of the labs and examples.

Unit Test

Unit Testing

Unit Test

A self-contained test where a part of the program is executed and matched against a known correct result

- Unit tests can be as big or as small as you like.
- The key is though, know what you are testing
- Your tests must obviously be correct! (Sometimes easier said than done!)

Unit Test - Example 1

```
1  #include <iostream>
2  // Some function
3  int square(int x){
4    return x*x;
5  }
6  // Simple unit test
7  bool unittest(int expected, int answer){
8    return (expected==answer);
9  }
10
11 int main(){
12   // Run some tests (1 means passed or true)
13   for(int i =0; i<10; ++i){
14     std::cout << "Test# " << i << "="<< unittest(i*i,square(i
       )) << "\n";
15   }
16   return 0;
17 }
```

Listing 1: Test our square function

Unit Test - Example 2

```
#include <iostream>
using namespace std;

// Unit test to see if two fixed-size arrays are sorted
bool correctlySorted(int* correct, int* myArray, int size){
  for(int i =0; i < size; ++i){
    if(correct[i]!=myArray[i]){
      return false;
    }
  }
  return true;
}
int main(){
  int sorted[] = {1,2,3,4,5};
  int unsorted[] = {2,4,3,5,1};
  std::cout << "Is it sorted: " << correctlySorted(sorted,
    unsorted,5) << '\n';
  return 0;
}
```

Listing 2: Check if an array is sorted

Unit Tests in Practice

- If you like breaking things, this can be your job.
- Many software engineers work as test engineers and do this exact thing.
- It is not uncommon to have thousands of tests (imagine how many there might be for the clang++ compiler!)
- And what is nice, is that as you update functionality to your program, you just run your test suite to see if it breaks any previous tests.
- The more code coverage you have(i.e. what % of functionality do you have a test for), the more confidence you can have in your software.

Assertions

Assertion

- Sometimes we want to make strong claims in our programs.
- If the claim is false for any reason, then the safest thing to do in the system is terminate execution.

Assertion

A statement that is always expected to be true.

Assertion

- Assertions are created with the *assert* function
- Assertions are checked while the program is running.
- We can use the C-preprocessor to turn on and off asserts if we do not want to test functionality.
- (In C++11 we can have static_assert, which is checked at compile-time)

Assert - Example 1

```
1  #include <iostream>
2  #include <cassert> // Adds in a MACRO assert function
3
4  int main(){
5
6      assert(1==1); // Expected to pass
7      assert(1==2); // Something is terribly wrong!!
8    std::cout << "Hello World\n";
9
10     return 0;
11 }
```

Listing 3: Check if the program runs

```
assert: asssert.cpp:7: int main(): Assertion `1==2' failed.
Aborted (core dumped)
```

Figure 5: Assertion is caught, and even tells us where!

Assert - Example 2

```
#include <iostream>
// Flips a switch in the compiler to basically ignore
// all assert statements. This goes before we include
// the cassert header file.
#define NDEBUG
#include <cassert> // Adds in a MACRO assert function

int main(){
    assert(1==1); // Expected to pass
    assert(1==2); // Something is terribly wrong!!
    std::cout << "Hello World\n";

    return 0;
}
```

Listing 4: assert statements ignored and Hello World is printed with a newline.

Assert effectiveness

- It is argued whether terminating a program right away is the best thing to do in the case of an error.
- I think asserts might be quite confusing to the user when running the software and it just shuts down.
- However, as a software developer, they are very helpful for testing your assumptions of the system.
- Let's say we want to do one step better, and actually handle errors as they occur.

Exceptions

Exception

- In some domains (say during our bank transactions), we do not want the default behavior to be to terminate the entire program if an error occurs.
- We may want to handle the error, log it, and then continue on with our the rest of our program.

Exception

A way to react to runtime errors in programs by transferring control to special functions called handlers.

Types of Exceptions

- A list is enumerated here:
  `http://en.cppreference.com/w/cpp/error/exception`
- These boil down to:
  - logic errors
  - runtime errors
  - bad data types
  - bad memory allocations
  - And bad function calls to name a few

Exception - Example

```
#include <iostream>

int main(){

    int age = -5;

    try{
        if(age < 0){
            throw age;
        }
    }catch(int e){
        // If we throw, then print an error message.
        std::cout << "Exception: throw age=" << e <<"\n";
    }

    std::cout << "Let's continue...\n";

    return 0;
}
```

Listing 5: Throw an integer exception

Testing Matters

- So as you have seen, testing matters!
- Lots of smart folks have put effort into building tools to automate some of this for us.
- A variety of Debuggers/IDEs/C++ Tools/Programmer Tools exist.
- Let's take a look at a few

Valgrind - Helps find memory errors

```
1  #include <iostream>
2
3  int main(){
4
5      int* myArray = new int[500];
6      myArray[0] = 0;
7      std::cout << myArray[0] << "\n";
8
9      // Oops, we forgot to free!
10     return 0;
11 }
```

Listing 6: The compiler will let us run this program

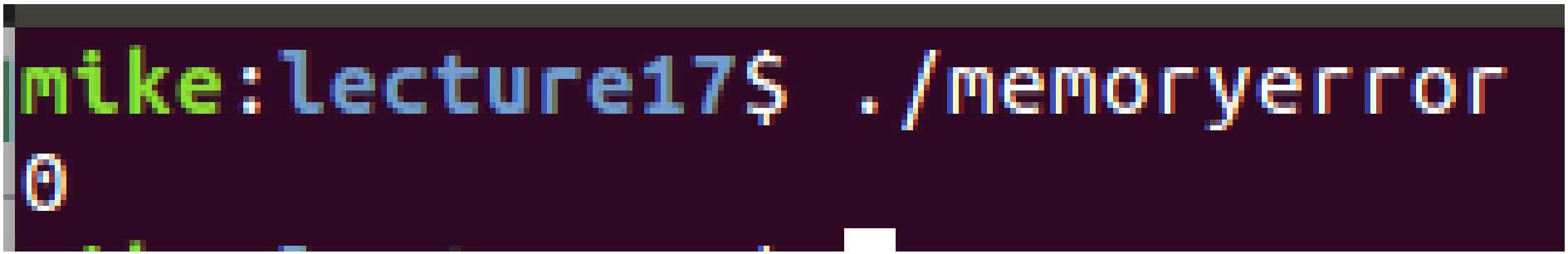


Figure 6: But we know something is going on!

Valgrind - Example

- Valgrind (available on CS machines or download) will hint where there are memory errors for us!

```
mike:lecture17$ valgrind --leak-check=full ./memoryerror
==29824== Memcheck, a memory error detector
==29824== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==29824== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==29824== Command: ./memoryerror
==29824==
0
==29824==
==29824== HEAP SUMMARY:
==29824==     in use at exit: 74,704 bytes in 2 blocks
==29824==   total heap usage: 3 allocs, 1 frees, 75,728 bytes allocated
==29824==
==29824== 2,000 bytes in 1 blocks are definitely lost in loss record 1 of 2
==29824==    at 0x4C2E80F: operator new[](unsigned long) (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==29824==    by 0x40089A: main (in /home/mike/Dropbox/AcademicWebsite/comp/11/lectures/lecture17/memoryerror)
```

Figure 7: Valgrind alerts us we failed to free blocks of memory!

GDB

- GDB is a debugger tool that allows you to step through a program.
- We can help GDB by adding a '-g' argument, which inserts special debugging symbols into our code for us, to allow us to control execution.
- example: clang++ -g main.cpp -o myProgram
- We will actually follow a tutorial for GDB for our in-class activity to get some hands on experience.

Integrated Development Environment (IDE)

- An Integrated Development Environment often has several of these tools built in.
- Some of the popular ones at this time for C++ are: Visual Studio(Windows), XCode(Mac), CodeBlocks(Unix), QtCreator(All platforms), Eclipse(All Platforms).
- Other features like intellisense and code completion are useful for productivity (these tools can be activated in VIM as well).

In-Class Activity

Part one of this tutorial! `https://aaronbloomfield.github.io/pdr/tutorials/02-gdb/index.html`

`http://www.mshah.io/comp/11/activities/activity15/activity.pdf`

Activity Discussion

Review of what we learned

- (At least) Two students
- Tell me each 1 thing you learned or found interesting in lecture.

5-10 minute break

To the lab!

Lab: `http://www.mshah.io/comp/11/labs/lab15/lab.pdf`
[2]

[2] You should have gotten an e-mail and hopefully setup an account at `https://www.eecs.tufts.edu/~accounts` prior to today. If not–no worries, we'll take care of it during lab!

Glossary

Assertion A statement that is always expected to be true.. 21

Exception . 27

Unit Test A self-contained test where a part of the program is executed and matched against a known correct result. 16