Debugging - Basic techniques and tools

Steven Costiou steven.costiou@inria.fr

RMoD / Inria Lille - Nord Europe

September 2022

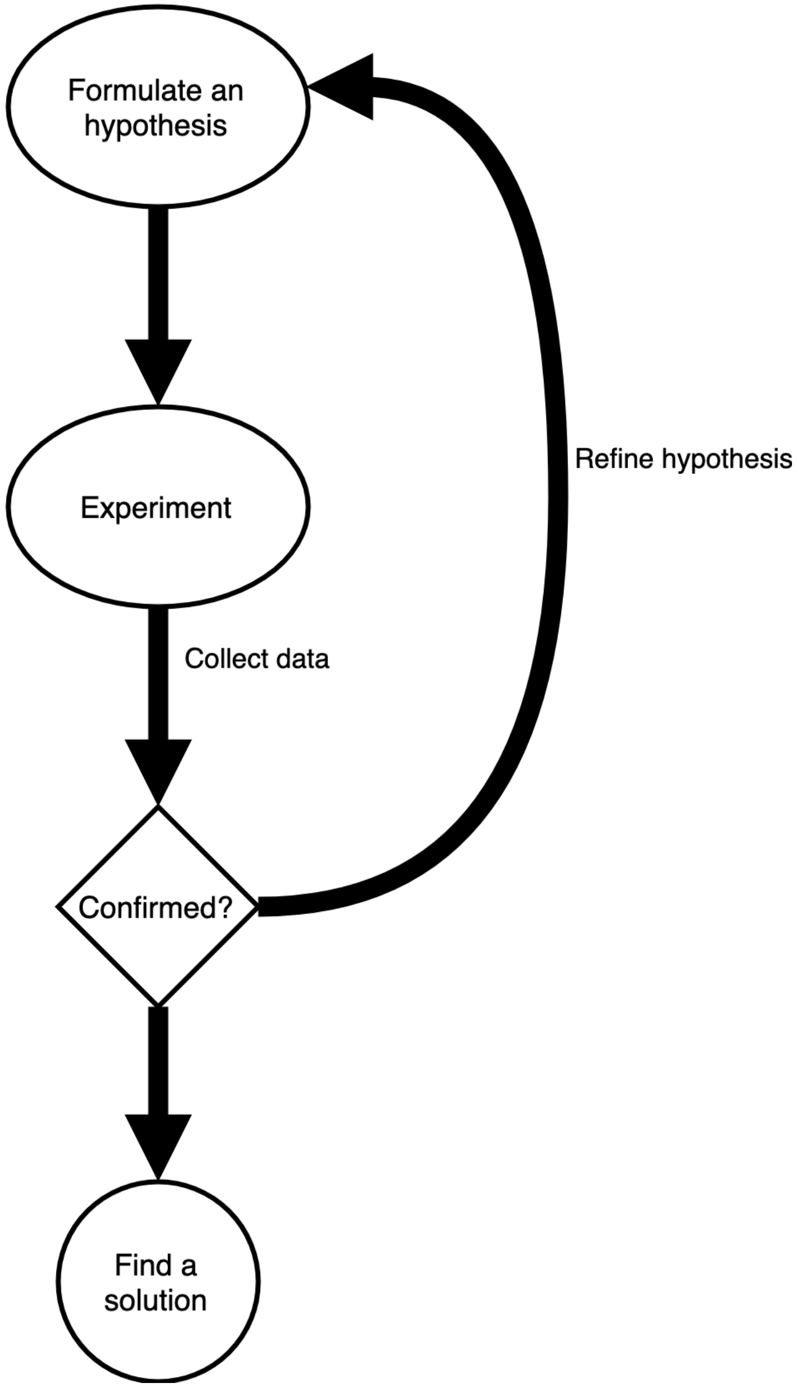
Understanding and fixing bugs?

- **This is not a recipe**
  - The following are general rules and advices
  - This is introductory to more reading
  - Those rules are complementary to practical experience

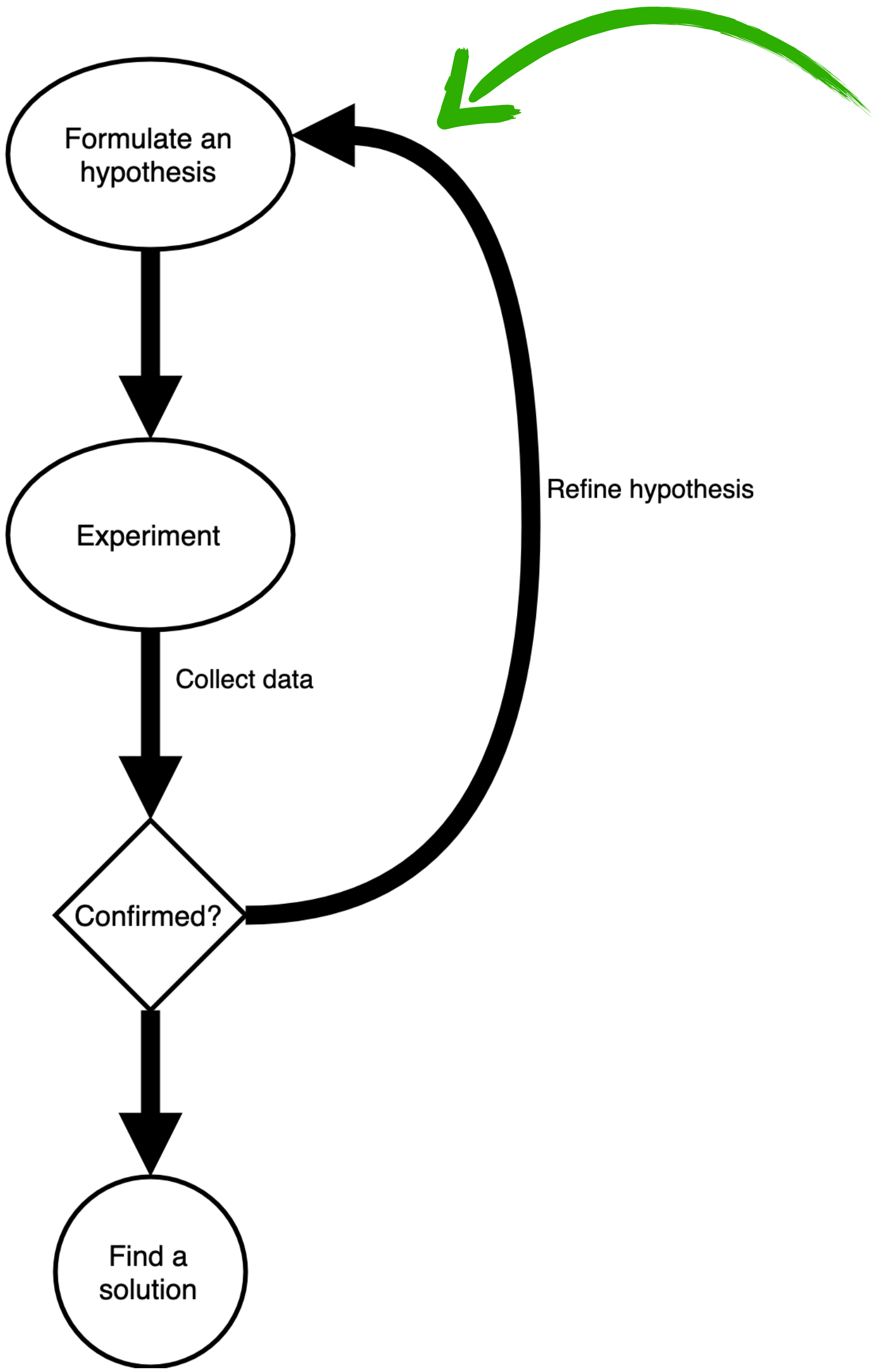
- **Good references for detailed methodology**

1. **Debugging: The 9 indispensable rules for finding even the most elusive software and hardware problems**, David J. Agans, 2002
2. **Why Programs Fail**, Andreas Zeller, 2009
3. **Effective Debugging**, Diomidis Spinellis, 2016
4. **The Science of Debugging**, Telles and Hsied, 2001

Overview: « simplified » simplified scientific method



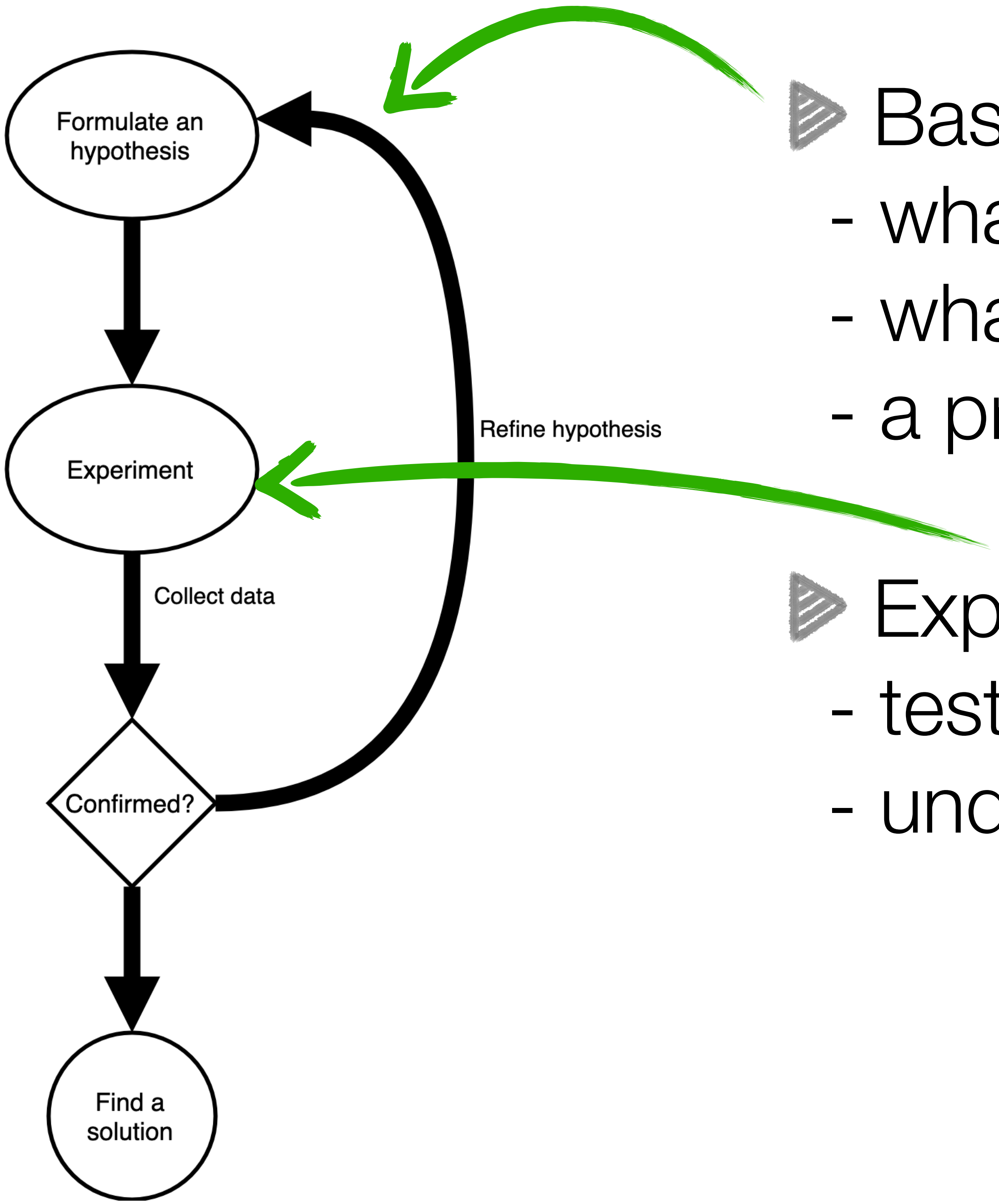
Overview: « simplified » simplified scientific method



Base your hypothesis on:

- what you observe
- what you know about the system
- a preliminary analysis of the system

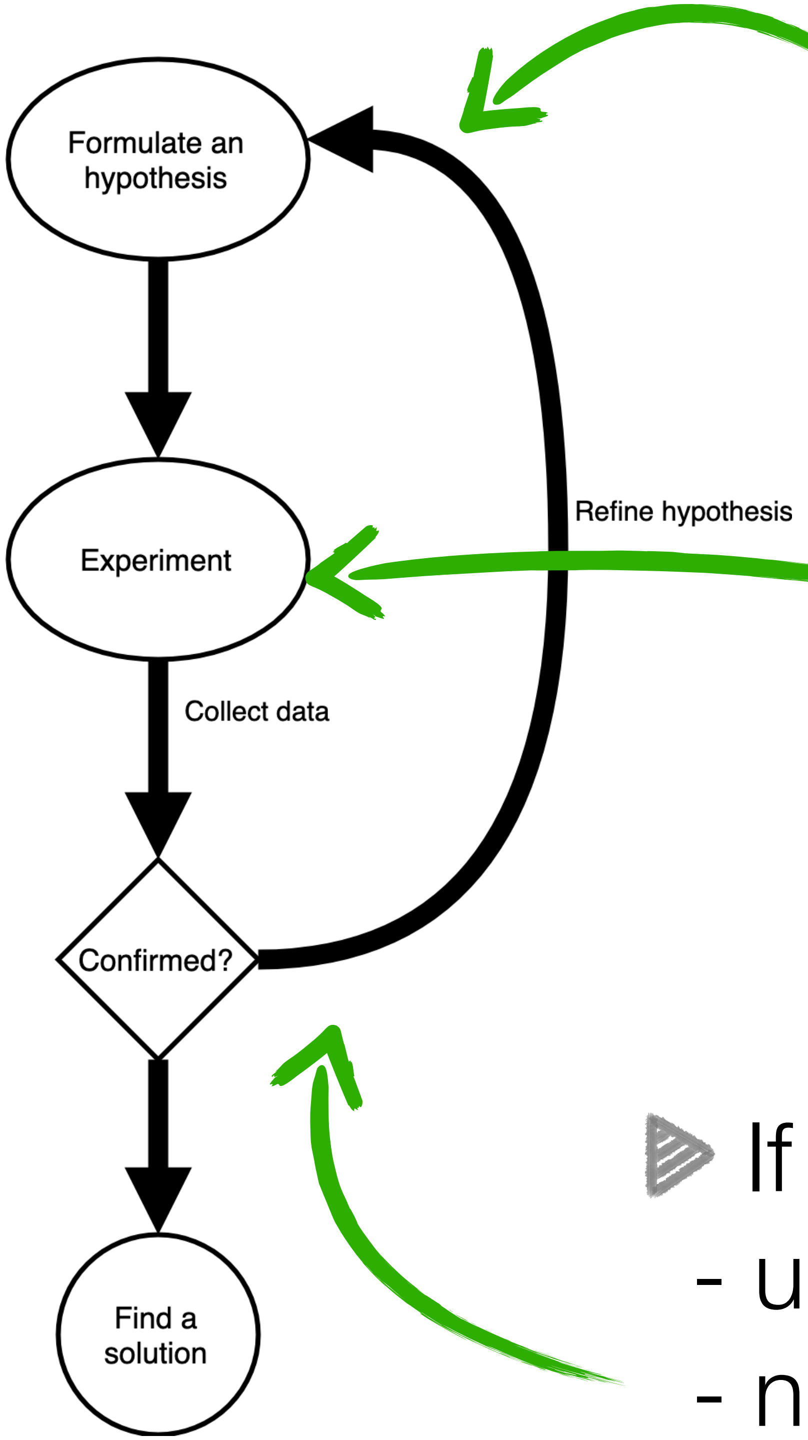
Overview: « simplified » simplified scientific method



- Base your hypothesis on:
  - what you observe
  - what you know about the system
  - a preliminary analysis of the system

- Experiment and collect data:
  - test your hypothesis correct
  - understand the system better

Overview: « simplified » simplified scientific method



▶ Base your hypothesis on:

- what you observe
- what you know about the system
- a preliminary analysis of the system

▶ Experiment and collect data:

- test your hypothesis correct
- understand the system better

▶ If your hypothesis is wrong or incomplete:

- use your new understanding to refine it
- narrow down your experimental setup to obtain more precise data

Overview: different steps

- **Observe the bug**
  - Make the program fail: reproduce the bug
  - Simplify the problem: reduce it to the smallest set of conditions
- **Narrow down the search**
  - Observe the smallest entity / reduce the search space
  - Use assertions to validate program state at key points of the program's execution
- **Fix the bug**
  - Write tests and execute tests suites to ensure non regression
  - Write assertions in the code to ensure the bug does not reappear

Observe the bug (1)

- **The objective is to control the program to force bug reproduction**
  - Reproduce the faulty environment
    - The bug may only happen in a specific context
    - The bug may happen in different environments (*e.g.*, production & dev)

Observe the bug (1)

- **The objective is to control the program to force bug reproduction**
  - Reproduce the faulty environment
    - The bug may only happen in a specific context
    - The bug may happen in different environments (*e.g.*, production & dev)
  - Reproduce the faulty execution
    - Control inputs: force the program to use specific values
    - Control behavior: force the program to use a specific mode or configuration
    - **This should not be random:** you must understand the program and formulate reasonable hypotheses about the possible problem's origin

Observe the bug (2)

- **Help yourself and others to reproduce the bug**
  - Write a step-by-step procedure
    - Applying the procedure guarantees bug reproduction
    - Usually in bug reports
    - Sometimes it is not possible to do otherwise

Observe the bug (2)

Help yourself and others to reproduce the bug

- Write a step-by-step procedure
  - Applying the procedure guarantees bug reproduction
  - Usually in bug reports
  - Sometimes it is not possible to do otherwise
- Write a unit test
  - Implement the minimal conditions for bug reproduction
  - Very powerful tool when it can be done
  - Executing the test reproduces the bug: you can observe it on demand!
  - This test should be included in the program's test suite and executed each time a change is done in the program (continuous integration, release…)

Narrow down the search (1)

- **Reduce the search space: you’re looking for the source of the bug**
  - Divide and conquer
    - Eliminate parts of the code not involved in the bug
    - Insert trace and instrumentation to infirm or confirm an hypothesis
    - Proceed by successive approximations

Narrow down the search (1)

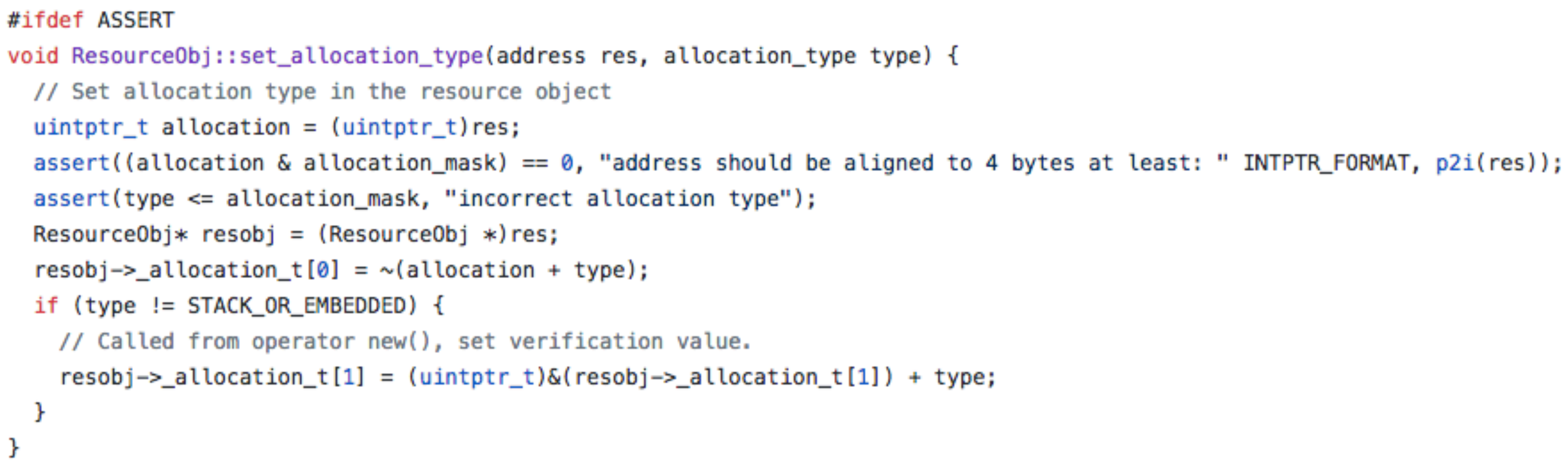
- **Reduce the search space: you're looking for the source of the bug**
  - Divide and conquer
    - Eliminate parts of the code not involved in the bug
    - Insert trace and instrumentation to infirm or confirm an hypothesis
    - Proceed by successive approximations
  - Use assertions
    - To validate key points of your program
    - To compare what is expected with what you observe

Narrow down the search (1)

- **Reduce the search space: you’re looking for the source of the bug**
  - Divide and conquer
    - Eliminate parts of the code not involved in the bug
    - Insert trace and instrumentation to infirm or confirm an hypothesis
    - Proceed by successive approximations
  - Use assertions
    - To validate key points of your program
    - To compare what is expected with what you observe
  - Target fine-grained entities
    - Use conditions to define your debugging space
      - Target variables and state involved in the bug
      - In object-oriented programs: debug objects (possibly only one)
    - Change one thing at a time: you need to be able to conclude information

Narrow down the search (2)

Example of assertion in the Java Virtual Machine (Open JDK)



```
#ifdef ASSERT
void ResourceObj::set_allocation_type(address res, allocation_type type) {
  // Set allocation type in the resource object
  uintptr_t allocation = (uintptr_t)res;
  assert((allocation & allocation_mask) == 0, "address should be aligned to 4 bytes at least: " INTPTR_FORMAT, p2i(res));
  assert(type <= allocation_mask, "incorrect allocation type");
  ResourceObj* resobj = (ResourceObj *)res;
  resobj->_allocation_t[0] = ~(allocation + type);
  if (type != STACK_OR_EMBEDDED) {
    // Called from operator new(), set verification value.
    resobj->_allocation_t[1] = (uintptr_t)&(resobj->_allocation_t[1]) + type;
  }
}
```

Fix the bug

- **When you find the bug, and fixed it**

Fix the bug

- **When you find the bug, and fixed it**
  - Explain the bug!
    - Answer the bug report
    - Organize a code review

Fix the bug

- **When you find the bug, and fixed it**
  - Explain the bug!
    - Answer the bug report
    - Organize a code review
  - Write tests!
    - Ensure that you can detect the bug if it reappears
    - Also counts as an explanation!

Fix the bug

- **When you find the bug, and fixed it**
  - Explain the bug!
    - Answer the bug report
    - Organize a code review
  - Write tests!
    - Ensure that you can detect the bug if it reappears
    - Also counts as an explanation!
- **You did not fix it…**
  - Because you cannot? This happens. Look for help.
  - Because it suddenly work? If you did nothing, it is still broken…

References

1. https://www.gnu.org/software/gdb/documentation/
2. http://kirste.userpage.fu-berlin.de/chemnet/use/info/gdb/gdb_8.html
3. http://cseweb.ucsd.edu/classes/fa09/cse141/tutorial_gcc_gdb.html
4. **Debugging with Gdb: The Gnu Source-level Debugger twelve Edition,** for Gdb Version, January 2018
5. **Debugging: The 9 indispensable rules for finding even the most elusive software and hardware problems,** David J. Agans, 2002
6. **Why Programs Fail,** Andreas Zeller, 2009
7. **Effective Debugging,** Diomidis Spinellis, 2016