The Abdus Salam
International Centre for Theoretical Physics

United Nations
Educational, Scientific and
Cultural Organization

IAEA
International Atomic Energy Agency

2068-20

Advanced School in High Performance and GRID Computing - Concepts and Applications

30 November - 11 December, 2009

Error messages and debugging

A. Kohlmeyer
University of Pennsylvania
Philadelphia
USA

Debugging

Advanced School in High Performance and GRID Computing

Axel Kohlmeyer

ICMS
Institute for Computational Molecular Science

TEMPLE UNIVERSITY®

What is Debugging?

- Identifying the cause of an error and correcting it
- Once you have identified defects, you need to:
  - find and understand the cause
  - remove the defect from your code
- Statistics show 60% of bug 'fixes' are not correct,
  -> they remove the symptom, but not the cause
- Improve productivity by getting it right the first time
- A lot of programmers don't know how to debug!
- Debugging needs practice and experience:
  -> understand the science and the tools

Debugging (2)

- Debugging is a last resort:
  - Doesn't add functionality
  - Doesn't improve the science
- The best debugging is to avoid bugs:
  - Good program design
  - Follow good programming practices
  - Always consider maintainability and readability of code over getting results fast
  - Maximize modularity and code re-use

Errors are Opportunities

- Learn from the program you're working on:
  - Errors mean you didn't understand the program,
    If you knew it perfectly, it wouldn't have an error.
    You would have fixed it already
- Learn about the kind of mistakes you make:
  - If you wrote the program, you inserted the error
  - Once you find a mistake, ask yourself:
    - Why did you make it?
    - How could you have found it more quickly?
    - How could you have prevented it?
    - Are there other similar mistakes in the code?

How to NOT do Debugging

- Find the error by guessing
- Change things randomly until it works again
- Don't keep track of what you changed
- Don't backup the original
- If the error is suddenly gone, trying to understand the problem, is a waste of time
- Fix the error with the most obvious fix
- If wrong code gives the correct result,
  and changing it doesn't work, don't correct it.

Debugging Tools

- Source code comparison tools:
  diff, vimdiff, tkdiff, emacs/ediff
  - Help you to find changes
- Source analysis tools:
  compiler warnings, ftnchek, lint
  - Help you to find problematic code
    -> Always enable warnings when programming
    -> Always take warnings seriously
    -> Always compile/test on multiple platforms
    -> Only ignore warnings you understand, if at all
- Debuggers: gdb, idb, pdbg, ddd (GUI)

Purpose of a Debugger

- More information than print statements
- Allows to stop/start/single step execution
- Look at data and modify it
- *'Post mortem'* analysis from core dumps
- Prove / disprove hypotheses
- Easier to use with modular code
- No substitute for good thinking
- But, sometimes good thinking is not a substitute for a good debugger!

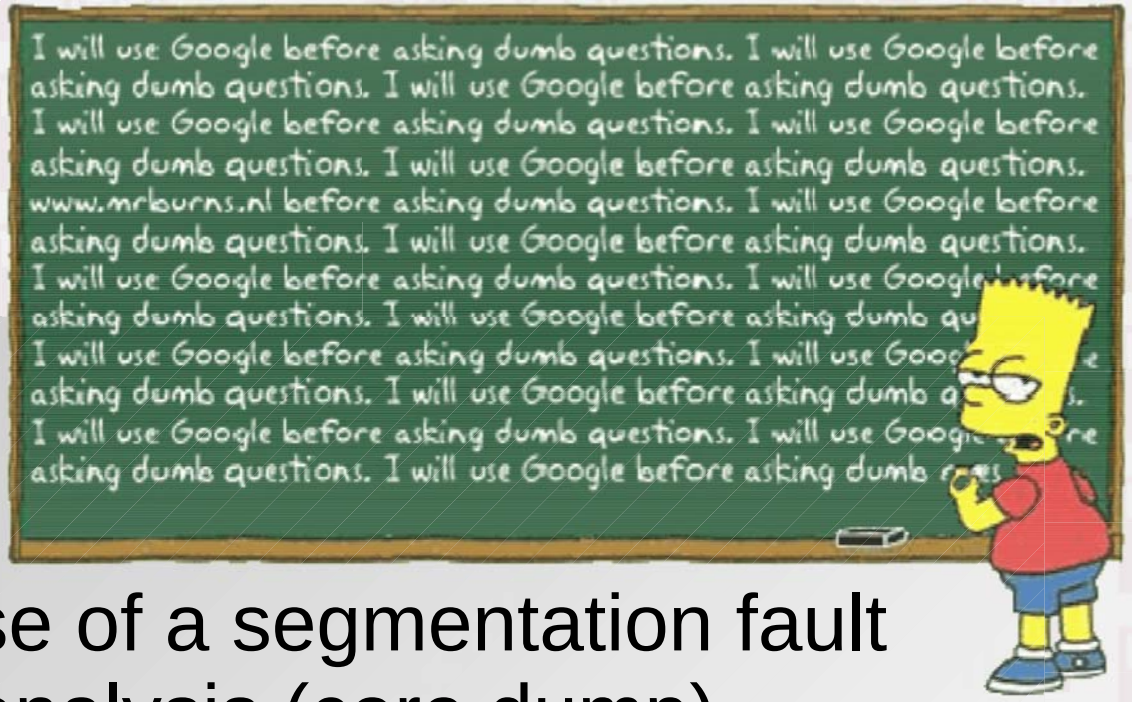
Using a Debugger

- When compiling use -g option to include debug info in object (.o) and executable
- 1:1 mapping of execution and source code only with disabled optimization
  -> problem when optimization uncovers bug
- GNU compilers allow -g with optimization
  -> not always correct line numbers
  -> variables/code can be 'optimized away'
- `strip` command removes debug info

How to Report a Bug

- Research whether bug is known/fixed
  -> web search, mailing list archive
- Provide description on how to reproduce the problem. Find a minimal input to show bug.
- Always state hardware/software you are using (distribution, compilers, appl. version)
- Demonstrate, that you have invested effort
- Make it easy for others to help you!

Demonstration



- Using a debugger. Available features.
- Identifying the cause of a segmentation fault from *post mortem* analysis (core dump).
- Identifying the cause of memory corruption from compiling with bounds checking
- Identifying memory leaks using `valgrind`
- `svn checkout --username akohlmey https://svn.gforge.escience-lab.org/svn/hpc-2008/trunk/week1/debugging`