Finding problems cheat sheet
Joel Fenwick, 2010
The University of Queensland

If your program does not compile see below. If your program does compile but does not work as expected see the diagram.

Program does not compile

Always deal with the first error and then recompile. The first thing to do is work out what type of error it is.

1. Compile error - your source code cannot be translated into object code.

    (a) Errors messages with line numbers are *mostly* compile errors.

    (b) Any sort of syntax error.

    (c) Invalid characters.

2. Linker error - your object code fragments can't be merged into a consistant program.

    (a) Unresolved symbols

    (b) Errors reported in .o files or files in /tmp (unless you put them there.)

    (c) The error does not appear when the `-c` switch is used.

Compile errors

- *Read the error message carefully.* If you do not understand what it is saying look it up or ask for help.

- Identify the line which the compiler is complaining about - be careful reading messages about files included by other files.
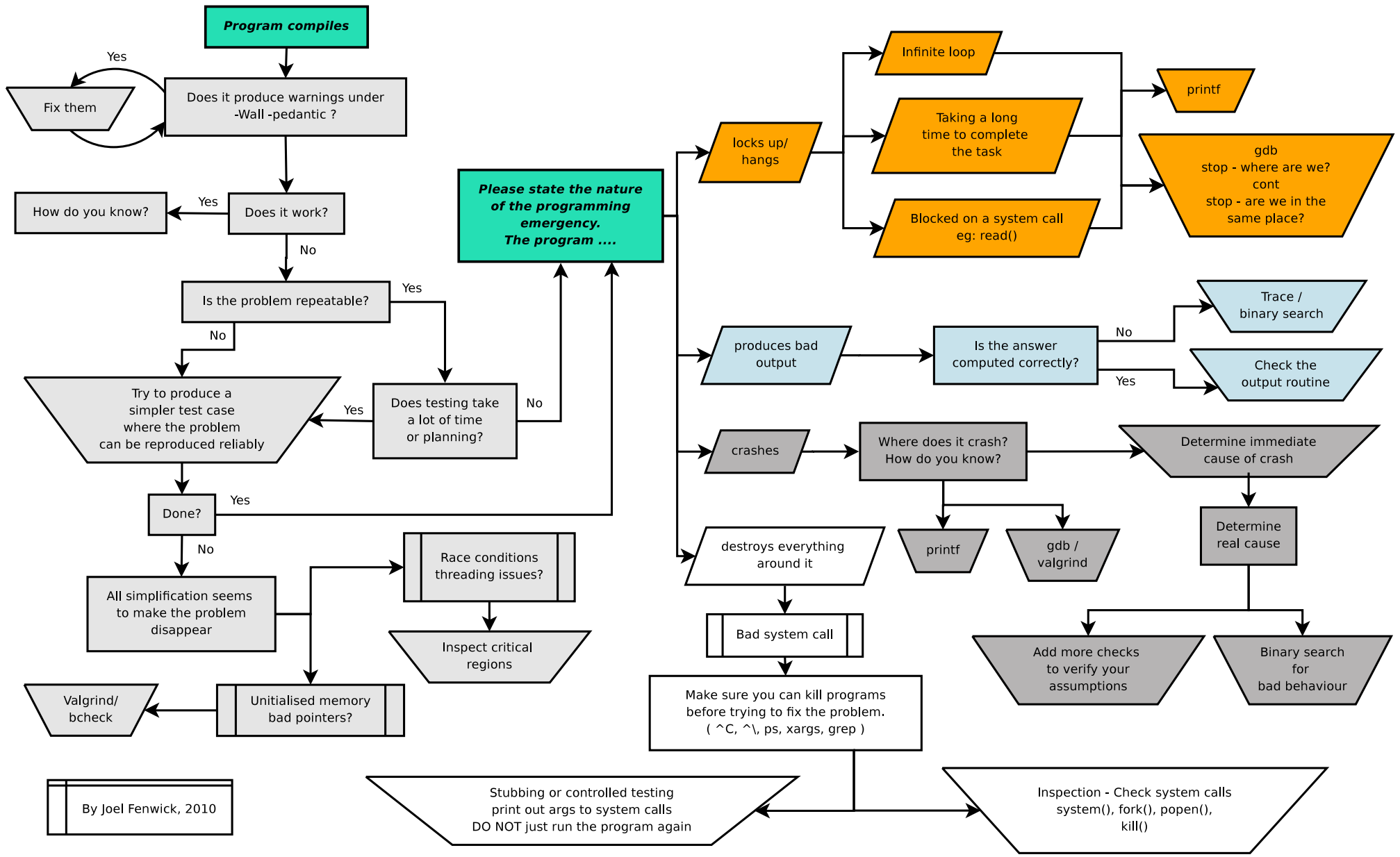
- Variables or functions being redeclared may indicate that you need `#ifndef` protection in your header files.

- Look on the previous line(s). A missing ; . or mismatched ([{ on earlier lines can confuse the compiler. A new line does not stop C from thinking of things as a unit.

- Do you have Dos/Windows end of lines in your files?

- If you have copy and pasted code from a pdf/webpage/other non-plain text document, make sure that the double quotes have not been replaced by open and close quotes.

Linker errors

- Have you listed all the libraries you need to link to? This is most likely to be a problem when you are writing a program using functions you have not called before. *Different operating systems have different library requirements - check the man pages for the functions you wish to call.*

- Not `#include`-ing a header where you use a function can confuse C.

- If a variable is included in a header make sure it is declared `extern`.

- Do not place function bodies in header files unless they are protected against duplicate definitions.

Misc

- Remember the goal is to understand problems and fix them. If a bug disappears and you do not understand what caused it, it may have moved somewhere else.

- When printing for debugging, use unbuffered output or flush your streams manually.

Program compiles

Does it produce warnings under -Wall -pedantic ?

Fix them — Yes

How do you know? ← Yes — Does it work?

No

Is the problem repeatable?

Yes → Does testing take a lot of time or planning?

No

Try to produce a simpler test case where the problem can be reproduced reliably

Yes ← (Does testing take a lot of time or planning?) No →

Done?

Yes →

No

All simplification seems to make the problem disappear

Race conditions threading issues?

Inspect critical regions

Unitialised memory bad pointers?

Valgrind/ bcheck

By Joel Fenwick, 2010

Please state the nature of the programming emergency. The program ....

locks up/ hangs

- Infinite loop
- Taking a long time to complete the task
- Blocked on a system call eg: read()

printf

gdb
stop - where are we?
cont
stop - are we in the same place?

produces bad output

Is the answer computed correctly?

No → Trace / binary search

Yes → Check the output routine

crashes

Where does it crash? How do you know?

printf

gdb / valgrind

Determine immediate cause of crash

Determine real cause

Add more checks to verify your assumptions

Binary search for bad behaviour

destroys everything around it

Bad system call

Make sure you can kill programs before trying to fix the problem.
(^C, ^\, ps, xargs, grep)

Stubbing or controlled testing
print out args to system calls
DO NOT just run the program again

Inspection - Check system calls
system(), fork(), popen(),
kill()