Working With Legacy Code

Las Vegas C/C++ Meetup Group - Tuesday August 14th, 2018

Presented by : Ray Imber

Preface / About me

- I am an experienced C programmer
- I don’t consider myself a C++ expert
- I am not a C/C++ “fanboy”
- But C/C++ is a practical language
- C/C++ is the industry standard for high performance computing

C/C++ is everywhere!

And therefore legacy code is everywhere

Where C/C++ is used

- C/C++ are industry standard for high performance Computing
- Video Games
  - Slot Machines
  - Computer Graphics for Movies
- Banking (industry in which I work)
- Aerospace / Automotive
- Operating Systems / Databases / Device Drivers / Compilers

Where C/C++ fits in the tech stack

- Usually sits “below” application software
    - Interface with Hardware (Driver)
    - A Database
    - A Web Browser
    - An Operating system
    - Compilers / Interpreters
    - An “Engine”
        - Java Virtual Machine (JVM)
        - A Computer Graphics engine (Unity / Unreal)

In Praise of Maintenance

How to work with Legacy Code

The Sad Truth about Software

- Most Software work involves maintainance http://freakonomics.com/podcast/in-praise-of-maintenance/
- Even if you are building new software, you will probably be using libraries
- Legacy Code is Archeology
  - You will be hunting down the source of bugs in large code bases that you did not write yourself

Real life Code is big!

https://informationisbeautiful.net/visualizations/million-lines-of-code/

- Quake 3 : ~400000 LOC
- Photoshop CS6 : ~5 Million LOC
- Firefox : ~ 10 Million LOC

You must use the principle of “Least Understanding”

Real life issues that occur in Legacy Code

- “code is legacy code as soon as it's written” - Programmer Proverb
  - “Hit by a bus” syndrome is real
- C and C++ have had lots of new features added in the past few years
  - This is great
  - Most code will not use these features
- Code documentation is wonderful
  - Much of the time it is not done well in practice
  - Especially in proprietary / closed source code

Tools of the Trade

What to look for and how to work with Legacy Code

Compilers

- Different Compilers will produce different code!
- Clang (Apple) doesn't support certain atomic operations natively
  - http://clang-developers.42468.n3.nabble.com/libc-help-diagnosing-the-following-std-atomic-compile-error-td4032210.html
- you can get linker errors because different compilers mangle C++ names differently
- Godbolt C++ explorer: cool tool that shows the assembly output from a C++ https://godbolt.org/

Build Systems

- Make
- CMake
- Meson
- QT
- Giant list on wikipedia: https://en.wikipedia.org/wiki/List_of_build_automation_software

Pro Tip: Learn Make and CMake! All the rest will follow!

Debuggers

- GDB
- LLDB
- Visual Studio Debugger

Pro Tip: Learn GDB and everything else will follow!

Basic GDB

- `gdb <path to program>`
- `run`
- `Ctrl-c` to interrupt program (stays in gdb)
- Set a breakpoint with `b filename:linenumber`

Core Dumps!!!

- On Linux, when a program crashes it will produce a “core dump” file
  - Must be enabled by setting `ulimit -c unlimited`
  - https://www.akadia.com/services/ora_enable_core.html
- `gdb <path to program> <path to core dump file>`
  - GDB will start, showing the state of the program when it crashed.
  - This is the bread and butter of debugging legacy code!
- The *-g* compiler flag
  - Do this if you can!
  - GDB will still work without it, but it is much harder

GDB on running programs (A.K.A magic!)

- `gdb -p <process id>`
  - Pauses a running program and shows you it's state inside GDB
  - Get the process id with *top*, *htop*, *or similar*

My favorite GDB Resources

- GDB Quick guides:
  - http://www.yolinux.com/TUTORIALS/GDB-Commands.html
  - http://condor.depaul.edu/glancast/373class/docs/gdb.html

- How GDB Works:
  - https://blog.0x972.info/?d=2014/11/13/10/40/50-how-does-a-debugger-work

Code Analysis Tips

How to read legacy code

Code with no Documentation

- “Self Documenting Code” is a lie
- Learn to reason about code with no documentation
- Read the function signature
- Get function examples from finding code that calls a function
- Follow the code like you were the cpu
  - Look up any function called that you don’t understand
    - This is recursive
  - Learn when to stop looking
    - Use Principle of Least Understanding

C Macros

- They are terrible
  - http://blog.robertelder.org/7-weird-old-things-about-the-c-preprocessor/
- Much legacy code still uses them
- Learn them
- Essentially automated copy / paste

The official GCC guide to Macros:
https://gcc.gnu.org/onlinedocs/cpp/Macros.html

C++ Templates

- Much better than C macros
- More complicated than C macros
- https://stackoverflow.com/questions/3766151/template-programming-common-mistakes
- Can make debugging / tracing difficult
  - Instantiated symbol of a template object cannot always be traced back to the template source

Exit(0);

Thanks!
Find me on LinkedIn @ https://www.linkedin.com/in/raymond-imber-3a445b21/
Find me on Github @ https://github.com/rayman22201