Chapter 16: The end of it all

July 22, 2016

Runtime environments

- Dynamic linking; indeed, in the Linux world, dynamic linking is even treated explicitly as a runtime interpreter
- Can otherwise be spare, like C's runtime
- Garbage collection (if it exists) is generally part of the runtime
- Variable argument library functions generally need some runtime help
- Exception handling (if it is supported) can be part of the runtime
- Event handling (like signal handling in the Unix world, or interrupt handling as might be found in an embedded environment) is almost always tied into the runtime

Might also be in the runtime environment

- RPCs?
- Transactional memory?

Very complete runtimes: virtual machines

- Virtual machines includes an abstract hardware environment
- Virtual machines can range from simple process VMs all the way through system VMs, such as QEMU

Process VMs are prevalent

- Lisp-family languages have some history of using a VM, such as SECD from the 1960s
- P-Code for Pascal was an early version back in the 1970s
- JVM in the 1990s had a lot of impact

Just-in-time (JIT) and dynamic compilation

- Powerful techniques (memoization of code rather than data, if you like)

Binary translation

- Difficult to make practical, but many projects have been done over the years;
- Generally not a long-term solution, but rather a transitional stopgap or an explicit part of virtual machine, such as QEMU

Inspection/Introspection

- While the use of metadata by debuggers and other infrastructure has a long history, there have also been a number of language implementations that allow introspection (or reflection) of metadata by an executing program
- Symbolic debugging is an old and common way of using such metadata: an external debugger starts the process that is to be studied.