C++ Runtime in the FreeBSD Kernel

A Work In Progress Talk by

ADAM David Alan Martin

adamartin@FreeBSD.org

Goals

- A loadable module “libc++.ko”
  - Developers wishing to write in C++ can depend upon this.
- Zero impact upon C development in the kernel.
- Full C++ language feature support
- Eventual full C++0x language feature support.

What does C++ provide?

- STL - Standard Template Library (lots of data structures and algorithms)
- Inheritance - Useful for related types
- Templates - Can be safer than macros
- Destructors/RAII - Safely manage resources
- Virtual functions - Easier dispatch-vectors
- Exceptions - Useful to report some errors.

Challenges and Tasks

- Choosing the right compiler options.
- Getting C++ STL libraries into the kernel.
- C++ runtime support for virtuals.
- C++ runtime support for exceptions.
- Getting kernel headers to behave nicely with C++. (There are some uses of C++ reserved words in them. Almost no headers have `extern "C"` blocks!)
- Licensing issues?

What’s done so far?

- Constructors
- Destructors
- Templates
- Inheritance (Including multiple and virtual)
- Virtual functions

What’s done so far? (continued)

- C++ new and delete map into malloc(9) and free(9), and support struct malloc.
- Many kernel resources have C++ RAII management classes, like struct mtx.
- Many STL components have been implemented.
  - Reimplemented under under a BSD license.

What’s not working?

- C++0x features -- We need compiler support first.
- C++ Exceptions. All of it! This is the hardest part of implementing a C++ runtime.
  - It currently panics when doing stack unwinding.
- C++ RTTI is not well supported.

Questions?

Don’t be shy!