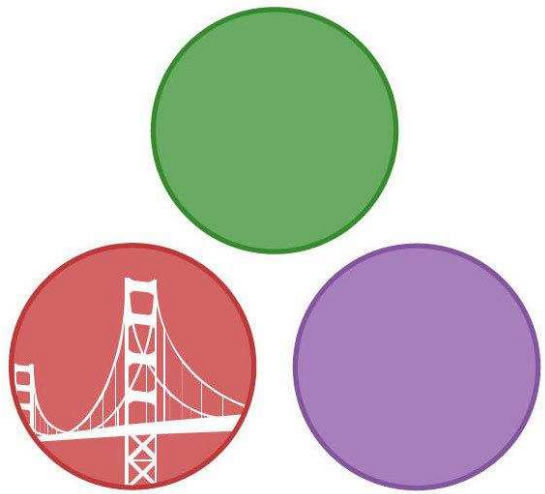
# A Guide to Julia's Dependencies

## aka why does Julia take so long to compile from source?

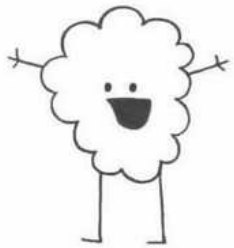
Tony Kelman

(@tkelman on Github)



Bay Area Julia Users
December 4, 2014

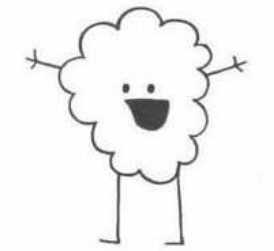
# Who's this guy?

- Grad student at Berkeley in Mechanical Engineering
- Julia user and contributor since February 2014
- On Github:  (look up "Rejected" on Youtube)
- Because I was curious, which files have I committed to most?

```
$ git log --author=kelman --pretty=oneline --name-only | sort | uniq -c | sort -r -n | head -n 15
     29 deps/Makefile
     17 Make.inc
     12 Makefile
      8 base/interactiveutil.jl
      6 src/flisp/Makefile
      6 README.md
      5 test/sparse.jl
      5 test/file.jl
      5 src/sys.c
      5 src/support/Makefile
      5 src/debuginfo.cpp
      5 LICENSE.md
      5 contrib/windows/msys_build.sh
      5 appveyor.yml
      5 .travis.yml
```

# Who's this guy?

- Grad student at Berkeley in Mechanical Engineering
- Julia user and contributor since February 2014



- On Github: (look up "Rejected" on Youtube)
- Because I was curious, which files have I committed to most?

```
$ git log --author=kelman --pretty=oneline --name-only | sort | uniq -c | sort -r -n | head -n 15
     29 deps/Makefile
     17 Make.inc
     12 Makefile
      8 base/interactiveutil.jl
      6 src/flisp/Makefile
      6 README.md
      5 test/sparse.jl
      5 test/file.jl
      5 src/sys.c
      5 src/support/Makefile
      5 src/debuginfo.cpp
      5 LICENSE.md
      5 contrib/windows/msys_build.sh
      5 appveyor.yml
      5 .travis.yml
```

subject of this talk

# Julia's dependencies

## Open source and the shoulders of giants

"If I have seen further it is by standing on the shoulders of giants." – Isaac Newton

"We all build our ideas on the best ideas we can find. Now imagine if there were no more good ideas we were allowed to use." – Bob Young, co-founder of Red Hat

- `git clone git://github.com/JuliaLang/julia.git`
  `cd julia && make` takes a while – the first time
- `deps/Makefile` downloads, configures, and compiles each dependency – a few of them are much bigger than Julia is
- Three groups of dependencies:
  1. Linked into `libjulia`, necessary to run the Julia JIT compiler
  2. Used in Julia's standard library via `ccall()` or process spawning
  3. Tools used only at build time to compile Julia (or other deps)

What does all this code do?

# Group 1, required for `libjulia`

- LLVM
  - JIT compiler
- libuv
  - Cross-platform input/output
- libunwind
  - Handling backtraces
- utf8proc
  - Unicode processing

# LLVM, http://llvm.org

- What: formerly "Low Level Virtual Machine," today general purpose compiler infrastructure
- Who: many contributors from Apple, Google, Intel, Mozilla, Julia, etc.
  Used by Clang, Rust, Swift, Emscripten, WebKit (Safari)
- When: originally Chris Lattner's Masters and Ph.D theses, circa 2003
- License: University of Illinois / NCSA (permissive, BSD-style)
- Written in: C++
- Use in Julia: just in time compiler

# A quick introduction to classical compiler design - 1

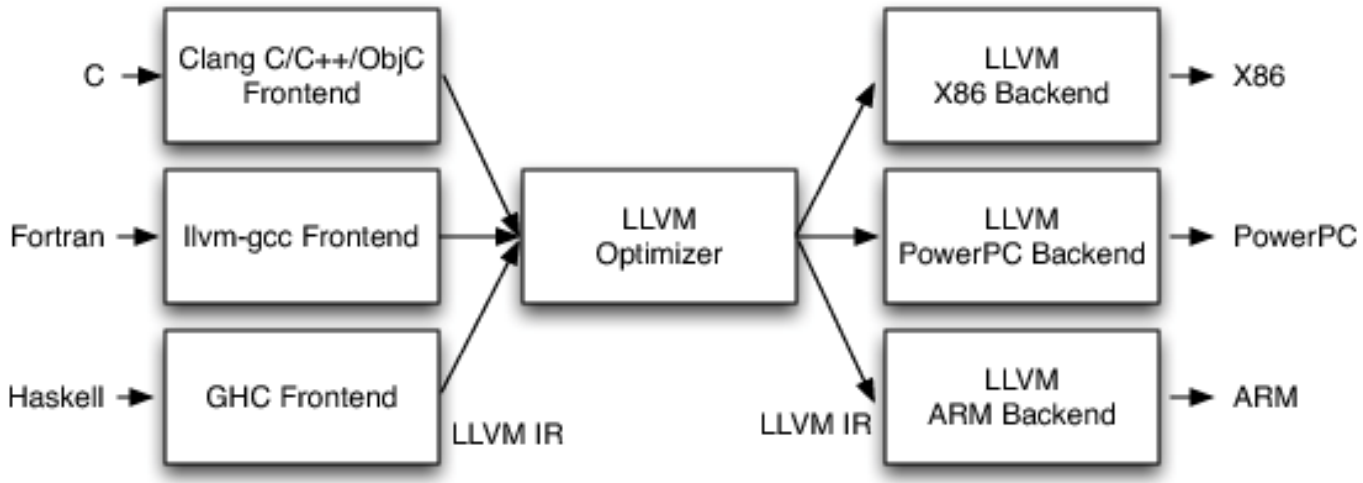
From "The Architecture of Open Source Applications," `http://www.aosabook.org/en/llvm.html`

- Basic 3 phase compiler


- One input language, one target architecture

# A quick introduction to classical compiler design - 2

- Modular compiler design




- Reuse core components across multiple languages and architectures
- LLVM intermediate representation (IR)
  - Sort of like "cross-platform assembly"
  - Try out `@code_llvm` in Julia

# libuv, https://github.com/libuv/libuv

- What: "Lib Unicorn Velociraptor"? pun on `-luv` to link to the library? cross platform asynchronous I/O
- Who: originally written to port Node.js to Windows, also used by Luvit, Pyuv, formerly Rust
- When: 2011
- License: MIT
- Written in: `C`
- Use in Julia: input/output, file system, process spawning, networking, terminal handling, without requiring POSIX

Julia uses a minor fork of libuv to support process piping syntax `run('ls'|>'sort')`, need work and pull requests to use upstream

# libunwind, http://nongnu.org/libunwind

- What: Support library for backtraces and profiling on Linux
- Who: David Mosberger, Arun Sharma, other contributors
- When: 2002
- License: MIT
- Written in: C

Different library from Apple, written in C++ and assembly, used on OSX
https://github.com/JuliaLang/libosxunwind

# utf8proc, http://public-software-group.org/utf8proc

- What: Unicode processing, normalization, case folding
- Who: Public Software Group
- When: 2006
- License: MIT
- Written in: C

Unicode is complicated!

- Variable width encodings
- Combining characters
- Different normalization forms
- Similar looking but distinct codepoints

Forked and updated to latest Unicode version by Steven G. Johnson and Jiahao Chen for Julia 0.4-dev, https://github.com/JuliaLang/libmojibake

# Group 2, used in Julia standard library

- OpenBLAS: Dense linear algebra
- SuiteSparse: Sparse matrix factorizations
- ARPACK: Sparse eigenvalue problems
- FFTW: Fourier transforms
- GMP: Arbitrary precision integers (`BigInt`)
- MPFR: Arbitrary precision floating point (`BigFloat`)
- PCRE: Regular expressions
- OpenLibm: Math functions normally found in `libm`
- OpenSpecFun: Additional special functions
- dSFMT: Random number generation
- Rmath: Distributions and statistical functions
- double-conversion: Floating point to string conversion (Julia $\leq$ 0.3.x)
- Git (in Julia $\leq$ 0.3.x) or libgit2 (0.4 WIP): Package manager

# OpenBLAS, http://openblas.net

| | | | | | |
|---|---|---|---|---|---|
| L | A | P | A | C | K |
| L | -A | P | -A | C | -K |
| L | A | P | A | -C | -K |
| L | -A | P | -A | -C | K |
| L | A | -P | -A | C | K |
| L | -A | -P | A | C | -K |

- What: Open source high performance implementation of the Basic Linear Algebra Subprograms (BLAS) and Linear Algebra Package (LAPACK)
- Who: originally Kazushige Goto, now maintained by Xianyi Zhang, Werner Saar, and others
- When: GotoBLAS 2002, OpenBLAS 2011
- License: BSD
- Written in: Fortran, assembly, C
- Use in Julia: dense linear algebra

Takes a long time to compile when `OPENBLAS_DYNAMIC_ARCH=1`, builds optimized kernel implementations for many recent CPU families (Nehalem, Sandy Bridge, Haswell, etc)

# BLAS in one slide

**Level 1 BLAS**

| | | |
|---|---|---|
| `SUBROUTINE xROTG ( A, B, C, S )` | Generate plane rotation | S, D |
| `SUBROUTINE xROTMG( D1, D2, A, B, PARAM )` | Generate modified plane rotation | S, D |
| `SUBROUTINE xROT ( N, X, INCX, Y, INCY, C, S )` | Apply plane rotation | S, D |
| `SUBROUTINE xROTM ( N, X, INCX, Y, INCY, PARAM )` | Apply modified plane rotation | S, D |
| `SUBROUTINE xSWAP ( N, X, INCX, Y, INCY )` | $x \leftrightarrow y$ | S, D, C, Z |
| `SUBROUTINE xSCAL ( N, ALPHA, X, INCX )` | $x \leftarrow \alpha x$ | S, D, C, Z, CS, ZD |
| `SUBROUTINE xCOPY ( N, X, INCX, Y, INCY )` | $y \leftarrow x$ | S, D, C, Z |
| `SUBROUTINE xAXPY ( N, ALPHA, X, INCX, Y, INCY )` | $y \leftarrow \alpha x + y$ | S, D, C, Z |
| `FUNCTION xDOT ( N, X, INCX, Y, INCY )` | $dot \leftarrow x^T y$ | S, D, DS |
| `FUNCTION xDOTU ( N, X, INCX, Y, INCY )` | $dot \leftarrow x^T y$ | C, Z |
| `FUNCTION xDOTC ( N, X, INCX, Y, INCY )` | $dot \leftarrow x^H y$ | C, Z |
| `FUNCTION xxDOT ( N, X, INCX, Y, INCY )` | $dot \leftarrow \alpha + x^T y$ | SDS |
| `FUNCTION xNRM2 ( N, X, INCX )` | $nrm2 \leftarrow \|x\|_2$ | S, D, SC, DZ |
| `FUNCTION xASUM ( N, X, INCX )` | $asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$ | S, D, SC, DZ |
| `FUNCTION IxAMAX( N, X, INCX )` | $amax \leftarrow 1^{st} k \ni \|re(x_k)\| + \|im(x_k)\| = max(\|re(x_i)\| + \|im(x_i)\|)$ | S, D, C, Z |

**Level 2 BLAS**

| | | |
|---|---|---|
| `xGEMV ( TRANS, M, N, ALPHA, A, LDA, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$ | S, D, C, Z |
| `xGBMV ( TRANS, M, N, KL, KU, ALPHA, A, LDA, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$ | S, D, C, Z |
| `xHEMV ( UPLO, N, ALPHA, A, LDA, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y$ | C, Z |
| `xHBMV ( UPLO, N, K, ALPHA, A, LDA, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y$ | C, Z |
| `xHPMV ( UPLO, N, ALPHA, AP, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y$ | C, Z |
| `xSYMV ( UPLO, N, ALPHA, A, LDA, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y$ | S, D |
| `xSBMV ( UPLO, N, K, ALPHA, A, LDA, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y$ | S, D |
| `xSPMV ( UPLO, N, ALPHA, AP, X, INCX, BETA, Y, INCY )` | $y \leftarrow \alpha Ax + \beta y$ | S, D |
| `xTRMV ( UPLO, TRANS, DIAG, N, A, LDA, X, INCX )` | $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$ | S, D, C, Z |
| `xTBMV ( UPLO, TRANS, DIAG, N, K, A, LDA, X, INCX )` | $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$ | S, D, C, Z |
| `xTPMV ( UPLO, TRANS, DIAG, N, AP, X, INCX )` | $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$ | S, D, C, Z |
| `xTRSV ( UPLO, TRANS, DIAG, N, A, LDA, X, INCX )` | $x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$ | S, D, C, Z |
| `xTBSV ( UPLO, TRANS, DIAG, N, K, A, LDA, X, INCX )` | $x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$ | S, D, C, Z |
| `xTPSV ( UPLO, TRANS, DIAG, N, AP, X, INCX )` | $x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$ | S, D, C, Z |
| `xGER ( M, N, ALPHA, X, INCX, Y, INCY, A, LDA )` | $A \leftarrow \alpha xy^T + A, A - m \times n$ | S, D |
| `xGERU ( M, N, ALPHA, X, INCX, Y, INCY, A, LDA )` | $A \leftarrow \alpha xy^T + A, A - m \times n$ | C, Z |
| `xGERC ( M, N, ALPHA, X, INCX, Y, INCY, A, LDA )` | $A \leftarrow \alpha xy^H + A, A - m \times n$ | C, Z |
| `xHER ( UPLO, N, ALPHA, X, INCX, A, LDA )` | $A \leftarrow \alpha xx^H + A$ | C, Z |
| `xHPR ( UPLO, N, ALPHA, X, INCX, AP )` | $A \leftarrow \alpha xx^H + A$ | C, Z |
| `xHER2 ( UPLO, N, ALPHA, X, INCX, Y, INCY, A, LDA )` | $A \leftarrow \alpha xy^H + y(\alpha x)^H + A$ | C, Z |
| `xHPR2 ( UPLO, N, ALPHA, X, INCX, Y, INCY, AP )` | $A \leftarrow \alpha xy^H + y(\alpha x)^H + A$ | C, Z |
| `xSYR ( UPLO, N, ALPHA, X, INCX, A, LDA )` | $A \leftarrow \alpha xx^T + A$ | S, D |
| `xSPR ( UPLO, N, ALPHA, X, INCX, AP )` | $A \leftarrow \alpha xx^T + A$ | S, D |
| `xSYR2 ( UPLO, N, ALPHA, X, INCX, Y, INCY, A, LDA )` | $A \leftarrow \alpha xy^T + \alpha yx^T + A$ | S, D |
| `xSPR2 ( UPLO, N, ALPHA, X, INCX, Y, INCY, AP )` | $A \leftarrow \alpha xy^T + \alpha yx^T + A$ | S, D |

**Level 3 BLAS**

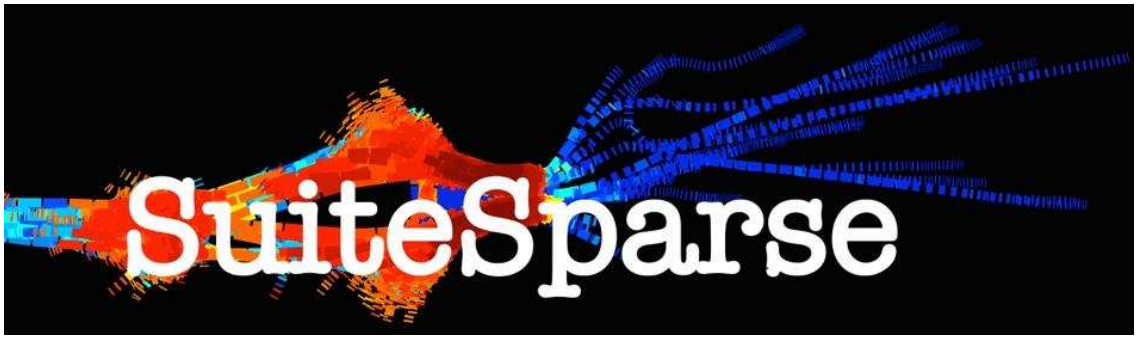
| | | |
|---|---|---|
| `xGEMM ( TRANSA, TRANSB, M, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC )` | $C \leftarrow \alpha op(A)op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$ | S, D, C, Z |
| `xSYMM ( SIDE, UPLO, M, N, ALPHA, A, LDA, B, LDB, BETA, C, LDC )` | $C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$ | S, D, C, Z |
| `xHEMM ( SIDE, UPLO, M, N, ALPHA, A, LDA, B, LDB, BETA, C, LDC )` | $C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$ | C, Z |
| `xSYRK ( UPLO, TRANS, N, K, ALPHA, A, LDA, BETA, C, LDC )` | $C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$ | S, D, C, Z |
| `xHERK ( UPLO, TRANS, N, K, ALPHA, A, LDA, BETA, C, LDC )` | $C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$ | C, Z |
| `xSYR2K( UPLO, TRANS, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC )` | $C \leftarrow \alpha AB^T + \bar{\alpha} BA^T + \beta C, C \leftarrow \alpha A^T B + \bar{\alpha} B^T A + \beta C, C - n \times n$ | S, D, C, Z |
| `xHER2K( UPLO, TRANS, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC )` | $C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C, C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C, C - n \times n$ | C, Z |
| `xTRMM ( SIDE, UPLO, TRANSA, DIAG, M, N, ALPHA, A, LDA, B, LDB )` | $B \leftarrow \alpha op(A)B, B \leftarrow \alpha Bop(A), op(A) = A, A^T, A^H, B - m \times n$ | S, D, C, Z |
| `xTRSM ( SIDE, UPLO, TRANSA, DIAG, M, N, ALPHA, A, LDA, B, LDB )` | $B \leftarrow \alpha op(A^{-1})B, B \leftarrow \alpha Bop(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$ | S, D, C, Z |

2

# Levels of BLAS and LAPACK

- Level 1 BLAS
  - Vector operations: scale, add, copy, dot product, find largest element
  - $O(n)$ operations on $O(n)$ data
- Level 2 BLAS
  - Matrix-vector operations: $A * x$, triangular solve, rank 1 or 2 updates
  - $O(n^2)$ operations on $O(n^2)$ data
- Level 3 BLAS
  - Matrix-matrix operations: $A * B$, triangular solve for multiple right hand sides, rank $k$ updates
  - $O(n^3)$ operations on $O(n^2)$ data
  - Where the important cache optimizations happen
- LAPACK (API does not fit on one slide)
  - Higher level factorizations, eigenvalue and singular value decompositions designed to use efficient BLAS-3 operations
- Reference Fortran implementations from Netlib are slow
  - Better to use an optimized (SIMD, multithreaded) implementation like OpenBLAS or Intel MKL

# SuiteSparse, `http://suitesparse.com`




- What: Sparse linear algebra – LU, Cholesky, and QR factorizations
- Who: Tim Davis, also used by Matlab, Mathematica, Google Ceres
- When: 2006 or earlier?
- License: LGPL
- Written in: `C`, `C++`

What makes a matrix sparse?

- Enough elements are zero to be worth taking advantage of
- Nonzero structure and permutations very important
- Graph theory for structure, linear algebra for numerics

# ARPACK, https://github.com/opencollab/arpack-ng

- What: ARnoldi PACKage for eigenvalue problems $Ax = \lambda x$ with sparse $A$, iterative algorithms to find a small set of $\lambda$ values
- Who: Rich Lehoucq, Kristi Maschoff, et al,
  also used by SciPy, Matlab, Octave
- When: 1996
- License: BSD
- Written in: Fortran 77

Work in progress to replace this with pure Julia code, see
https://github.com/JuliaLang/IterativeSolvers.jl/pull/31

# FFTW, http://fftw.org


- What: Fastest Fourier Transform in the West
- Who: Matteo Frigo and Steven G. Johnson
- When: 1997
- License: GPL
- Written in: C, code generator in OCaml

Work in progress to write a pure Julia FFT and move FFTW to an optional package, see https://github.com/JuliaLang/julia/pull/6193

# GMP, https://gmplib.org


- What: GNU Multiple Precision library for arbitrary precision integer arithmetic (`BigInt` in Julia)
- Who: Torbjörn Granlund, GNU Project, many contributors
- When: 1991
- License: LGPL
- Written in: `C`, assembly

# MPFR, `http://mpfr.org`


- What: GNU Multiple Precision Floating-point Reliable library for arbitrary precision floating-point arithmetic (`BigFloat` in Julia)
- Who: INRIA, GNU Project, many contributors
- When: 2000
- License: LGPL
- Written in: `C`

# PCRE, `http://pcre.org`

> Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems. – Jamie Zawinski

- What: Perl Compatible Regular Expressions (`r"..."` in Julia)
- Who: Philip Hazel, Zoltan Herczeg
- When: 1997
- License: BSD
- Written in: `C`

# OpenLibm, `http://openlibm.org`

- What: "high quality, portable, standalone C mathematical library"
- Who: Viral Shah and other JuliaLang contributors, code originally from FreeBSD msun, OpenBSD libm, and Freely Distributable FDLIBM (`http://www.netlib.org/fdlibm`)
- When: 2011, original code from 1992 or earlier
- License: BSD
- Written in: `C`, assembly
- Why: Performance and accuracy of system `libm` for trig, sqrt, exp, log, etc varies between platforms, more reliable to build our own

# OpenSpecFun, https://github.com/JuliaLang/openspecfun

- What: Collection of special functions – Bessel and Airy functions from AMOS library, complex error functions from Faddeeva
- Who: Donald Amos from http://netlib.org/amos, Faddeeva by Steven G. Johnson http://ab-initio.mit.edu/wiki/index.php/Faddeeva_Package
- When: Amos code from 1985 or earlier, Faddeeva 2012, OpenSpecFun 2013
- License: MIT, Public Domain
- Written in: Fortran 77, C++, C
- Why: Pieces that were split out from OpenLibm because they are not included in system libm libraries

# dSFMT, http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT

- What: Double precision SIMD-oriented Fast Mersenne Twister random number generator (RNG)
- Who: Matsuo Saito and Makoto Matsumoto
- When: 2007
- License: BSD
- Written in: C

Julia's RNG code is undergoing heavy development on 0.4-dev right now, good chance dSFMT might be replaced by some alternate RNG
see https://github.com/JuliaLang/julia/issues/8786

# Rmath, https://github.com/JuliaLang/Rmath

- What: Distributions and statistical functions from R, forked and patched to use same dSFMT RNG as Julia
- Who: R contributors, fork by Viral Shah
- When: Possibly as old as R, 1997? JuliaLang fork from 2013
- License: GPL
- Written in: C

Not used by base Julia, only Distributions.jl and HypothesisTests.jl!
see https://github.com/JuliaStats/Distributions.jl/pull/138 for work to remove the dependency

# double-conversion aka Grisu,

https://github.com/floitsch/double-conversion

- What: Efficient conversion between floating point and shortest equivalent decimal representation
- Who: Florian Loitsch, used by Google in V8
- When: 2010
- License: BSD
- Written in: C++

No longer used in Julia 0.4-dev, algorithm was ported to Julia by Jacob Quinn in https://github.com/JuliaLang/julia/pull/7291

# Git, http://git-scm.com


- What: Version control for Julia's package manager
- Who: Linus Torvalds, Junio Hamano, many contributors
- When: 2005
- License: GPL
- Written in: C, shell, Perl

WIP to switch to libgit2 (https://libgit2.github.com) instead of command-line executable for Julia 0.4-dev to improve speed of Pkg, speak up at https://github.com/JuliaLang/julia/issues/7584 if you want to help!

# Group 3, used during build process

- Patchelf
  - Setting `rpath` on Linux, `http://nixos.org/patchelf.html`
- Objconv
  - Renaming library symbols on OSX, `http://www.agner.org/optimize/#objconv`
- Virtualenv
  - Sandboxing Sphinx version for building documentation, `https://virtualenv.pypa.io`

Not much more to say about these

# That's it!

Hopefully you have a better sense what's going on,
next time you compile Julia from source.

Any questions?