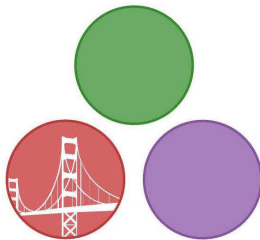


A Guide to Julia's Dependencies

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



(look up "Rejected" on Youtube)

- On Github:
- 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



(look up "Rejected" on Youtube)

- On Github:
- 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



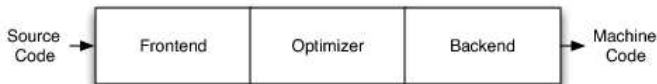
- What: formerly “Low Level Virtual Machine,” today general purpose compiler infrastructure
- Who: many contributors from Apple, Google, Intel, Mozilla, Julia, etc. Used by Clang, Rust, Swift, Emscripten, WebKit (Safari)
- When: originally Chris Lattner’s Masters and Ph.D theses, circa 2003
- License: 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>

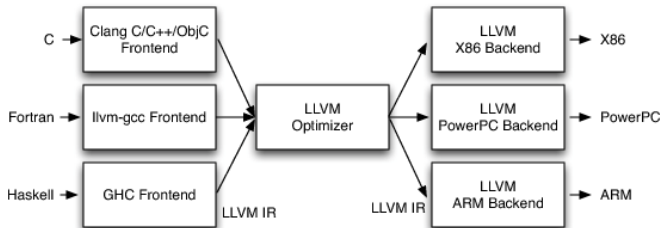
- Basic 3 phase compiler



- One input language, one target architecture

A quick introduction to classical compiler design - 2

- Modular compiler architecture



- 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

| | dim | scalar | vector | vector | scalars | 5-element array | |
|-----------------------|-----|--------|-------------------|--------|---------------|-----------------|--|
| SUBROUTINE xROTG (| | | | | A, B, C, S) | | |
| SUBROUTINE xROTNG (| | | | | D1, D2, A, B, | PARAM) | |
| SUBROUTINE xROT (N, | | | X, INCX, Y, INCY, | | C, S) | | |
| SUBROUTINE xROTH (N, | | | X, INCX, Y, INCY, | | | PARAM) | |
| SUBROUTINE xSWAP (N, | | | X, INCX, Y, INCY) | | | | |
| SUBROUTINE xSCAL (N, | | ALPHA, | X, INCX) | | | | |
| SUBROUTINE xCOPY (N, | | | X, INCX, Y, INCY) | | | | |
| SUBROUTINE xALPY (N, | | ALPHA, | X, INCX, Y, INCY) | | | | |
| FUNCTION xDOT (N, | | | X, INCX, Y, INCY) | | | | |
| FUNCTION xDOTU (N, | | | X, INCX, Y, INCY) | | | | |
| FUNCTION xDOTC (N, | | | X, INCX, Y, INCY) | | | | |
| FUNCTION xzDOT (N, | | | X, INCX, Y, INCY) | | | | |
| FUNCTION xNMZ (N, | | | X, INCX) | | | | |
| FUNCTION xASUM (N, | | | X, INCX) | | | | |
| FUNCTION IxAMAX (N, | | | X, INCX) | | | | |

Generate plane rotation
 Generate modified plane rotation
 Apply plane rotation
 Apply modified plane rotation
 $x \leftrightarrow y$
 $x \leftarrow \alpha x$
 $y \leftarrow x$
 $y \leftarrow \alpha x + y$
 $\text{dot} \leftarrow x^T y$
 $\text{dot} \leftarrow x^T y$
 $\text{dot} \leftarrow x^H y$
 $\text{dot} \leftarrow \alpha + x^T y$
 $\text{norm2} \leftarrow \|x\|_2$
 $\text{asum} \leftarrow |re(x)|_1 + |im(x)|_1$
 $\text{amax} \leftarrow 1^{st} k \ni |re(x_k)| + |im(x_k)|$
 $\quad = \max(|re(x_i)| + |im(x_i)|)$

prefixes
 S, D
 S, D
 S, D
 S, D
 S, D, C, Z
 S, D, C, Z, CS, ZD
 S, D, C, Z
 S, D, C, Z
 S, D, DS
 C, Z
 C, Z
 SDS
 S, D, SC, DZ
 S, D, SC, DZ
 S, D, C, Z

Level 2 BLAS

| | options | dim | b-width | scalar | matrix | vector | scalar | vector |
|---------|--------------------|---------------|---------|-----------------|------------------|----------------|--------|--------|
| xGENV (| TRANS, | M, N, | | ALPHA, A, LDA, | X, INCX, | BETA, Y, INCY) | | |
| xHEMV (| UPLO, TRANS, | M, N, KL, KU, | | ALPHA, A, LDA, | X, INCX, | BETA, Y, INCY) | | |
| xHEMV (| UPLO, | M, N, | | ALPHA, A, LDA, | X, INCX, | BETA, Y, INCY) | | |
| xHPMV (| UPLO, | M, K, | | ALPHA, A, LDA, | X, INCX, | BETA, Y, INCY) | | |
| xSYMV (| UPLO, | M, N, | | ALPHA, A, LDA, | X, INCX, | BETA, Y, INCY) | | |
| xSBMV (| UPLO, | M, K, | | ALPHA, A, LDA, | X, INCX, | BETA, Y, INCY) | | |
| xSPMV (| UPLO, | M, N, | | ALPHA, AP, | X, INCX, | BETA, Y, INCY) | | |
| xTRMV (| UPLO, TRANS, DIAG, | M, N, | | A, LDA, | X, INCX) | | | |
| xTBMV (| UPLO, TRANS, DIAG, | M, K, | | A, LDA, | X, INCX) | | | |
| xTPMV (| UPLO, TRANS, DIAG, | M, N, | | AP, | X, INCX) | | | |
| xTRSV (| UPLO, TRANS, DIAG, | M, N, | | A, LDA, | X, INCX) | | | |
| xTSBV (| UPLO, TRANS, DIAG, | M, K, | | A, LDA, | X, INCX) | | | |
| xTSBV (| UPLO, TRANS, DIAG, | M, N, | | AP, | X, INCX) | | | |
| xGER (| | M, N, | | ALPHA, X, INCX, | Y, INCY, A, LDA) | | | |
| xGERU (| | M, N, | | ALPHA, X, INCX, | Y, INCY, A, LDA) | | | |
| xGERC (| | M, N, | | ALPHA, X, INCX, | Y, INCY, A, LDA) | | | |
| xHER (| UPLO, | M, N, | | ALPHA, X, INCX, | A, LDA) | | | |
| xHPR (| UPLO, | M, N, | | ALPHA, X, INCX, | AP) | | | |
| xHER2 (| UPLO, | M, N, | | ALPHA, X, INCX, | Y, INCY, A, LDA) | | | |
| xHPR2 (| UPLO, | M, N, | | ALPHA, X, INCX, | Y, INCY, AP) | | | |
| xSYR (| UPLO, | M, N, | | ALPHA, X, INCX, | A, LDA) | | | |
| xSPR (| UPLO, | M, N, | | ALPHA, X, INCX, | AP) | | | |
| xSYR2 (| UPLO, | M, N, | | ALPHA, X, INCX, | Y, INCY, A, LDA) | | | |
| xSPR2 (| UPLO, | M, N, | | ALPHA, X, INCX, | Y, INCY, AP) | | | |

$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$
 $y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$
 $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$
 $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$
 $x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$
 $x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$
 $x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$

S, D, C, Z
 S, D, C, Z
 C, Z
 C, Z
 C, Z
 S, D
 S, D
 S, D
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z

$A \leftarrow \alpha xy^T + A, A - m \times n$
 $A \leftarrow \alpha xy^T + A, A - m \times n$
 $A \leftarrow \alpha xy^H + A, A - m \times n$
 $A \leftarrow \alpha xy^H + A$
 $A \leftarrow \alpha xy^H + A$
 $A \leftarrow \alpha xy^H + y(\alpha x)^H + A$
 $A \leftarrow \alpha xy^H + y(\alpha x)^H + A$
 $A \leftarrow \alpha x x^T + A$
 $A \leftarrow \alpha x x^T + A$
 $A \leftarrow \alpha x x^T + A$
 $A \leftarrow \alpha xy^T + \alpha y x^T + A$
 $A \leftarrow \alpha xy^T + \alpha y x^T + A$

S, D
 C, Z
 C, Z
 C, Z
 C, Z
 C, Z
 C, Z
 S, D
 S, D
 S, D
 S, D

Level 3 BLAS

| | options | dim | scalar | matrix | matrix | scalar | matrix |
|---------|---------------------|--------------|--------|----------------|---------------|---------------|--------|
| xGEMM (| TRANSA, TRANSB, | M, N, K, | | ALPHA, A, LDA, | B, LDB, | BETA, C, LDC) | |
| xSYMM (| SIDE, UPLO, | M, N, | | ALPHA, A, LDA, | B, LDB, | BETA, C, LDC) | |
| xHEMM (| SIDE, UPLO, | M, N, | | ALPHA, A, LDA, | B, LDB, | BETA, C, LDC) | |
| xSYRK (| UPLO, TRANS, | M, K, ALPHA, | | A, LDA, | BETA, C, LDC) | | |
| xHERK (| UPLO, TRANS, | M, K, ALPHA, | | A, LDA, | BETA, C, LDC) | | |
| xSYRK (| UPLO, TRANS, | M, K, ALPHA, | | A, LDA, | B, LDB, | BETA, C, LDC) | |
| xHERK (| UPLO, TRANS, | M, K, ALPHA, | | A, LDA, | B, LDB, | BETA, C, LDC) | |
| xTRMM (| SIDE, UPLO, TRANSA, | DIAG, M, N, | | ALPHA, A, LDA, | B, LDB) | | |
| xTRSM (| SIDE, UPLO, TRANSA, | DIAG, M, N, | | ALPHA, A, LDA, | B, LDB) | | |

$C \leftarrow \alpha op(A) op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$
 $C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$
 $C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$
 $C \leftarrow \alpha A A^T + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$
 $C \leftarrow \alpha A A^H + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$
 $C \leftarrow \alpha A B^T + \beta C, C \leftarrow \alpha A^T B + \beta B^T A + \beta C, C - n \times n$
 $C \leftarrow \alpha A B^H + \beta C, C \leftarrow \alpha A^H B + \beta B^H A + \beta C, C - n \times n$
 $B \leftarrow \alpha op(A) B + \alpha B op(A), op(A) = A, A^T, A^H, B - m \times n$
 $B \leftarrow \alpha op(A^{-1}) B + \alpha B op(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$

S, D, C, Z
 S, D, C, Z
 C, Z
 S, D, C, Z
 C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z

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://suitsparse.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 λ values
- Who: Rich Lehoucq, Kristi Maschhoff, 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>



- 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>



- 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



- 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

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?