

Death {by,to} Dynamic Linking

GHC 7.8

- System dynamic linker for GHCi
- Why?
 - building our own linker is hard
 - OS X changes its object format
 - some features (were) not supported (weak symbols, constructors/destructors)
 - problems with using some external libraries (C++?) in GHCi

Implications

- Building code for dynamic linking is a different "way"
- Cabal must build stuff both ways
 - So that we can use packages in GHCi
- Concerns about overhead and backwards-compat
 - static linking is still the default for GHC
- We added `-dynamic-too` to reduce the cost of building dynamic
- Need to link shared libs on the fly in GHCi to load compiled code

What happened

- Some things work in GHCi that didn't before
- We can get GHCi support on some platforms where we didn't before (using LLVM backend + dynamic linking or via C)
- GHCi starts up faster
- Fewer weird things in the base package to support having two copies of base.

Fallout

- If you use TH, we need dynamic objects, so -dynamic-too is enabled automatically (slower compilation)
- Still doesn't work on Windows (GHC package too big)
- Complication in the compiler to support -fPIC/-dynamic
- Cabal must build both versions, takes 2x as long
- -dynamic-too is still slower than -static
- Had to drop -dynamic optimisation that makes intra-package calls fast
- bugs:
 - GHCi doesn't pick up -dynamic-too objects
 - Interrupting -dynamic-too compilations leaves things in a weird state
 - still need the RTS linker (perhaps only for x86_64?)