Shake: A Better Make

Neil Mitchell, Standard Chartered
Haskell Implementors Workshop 2010
Question

Do you like your Make system?

- Make system builds multi-language stuff
  - Not ghc --make, cabal install, Visual Studio
import Development.Shake
main = shake $ do
  want ["Main.exe"]
  "Main.exe" *> \x -> do
      cs <- ls "*.c"
    let os = map (\replaceExtension\ "obj") cs
    need os
    system $ ["gcc","-o",x] ++ os
  "*.obj" *> \x -> do
      let c = replaceExtension x "c"
    need [c]
    need =<< cIncludes c
    system ["gcc","-c",c,"-o",x]
Benefits of Shake

- A Haskell library for writing build systems
  - Can use modules/functions for abstraction/separation
  - Can use Haskell libraries (i.e. filepath)

- It’s got the useful bits from Make
  - Automatic parallelism
  - Minimal rebuilds

- But it’s better!
  - More accurate dependencies (i.e. the results of ls are tracked)
  - Can produce profiling reports (what took most time to build)
  - Can deal with generated files properly
  - Properly cross-platform
Quick Tour

shake :: Shake () -> IO ()

want :: [FilePath] -> Shake ()
(*>) :: String -> (FilePath -> Act ()) -> Shake ()

need :: [FilePath] -> Act ()

system :: [String] -> Act ()

ls :: String -> Act [FilePath]
Sugar Functions

readFileLines :: FilePath -> Act [String]
readFileLines x = do
    need [x]
    liftIO $ fmap lines $ readFile x

copy :: FilePath -> FilePath -> Act()
copy from to = do
    mkdir $ takeDirectory to
    need [from]
    system' ["cp",quote from,quote to]
The Oracle is used for non-file dependencies
- What is the version of GHC? 6.12.3
- What extra flags do we want? --Wall
- ls is a sugar function for the Oracle

type Question = (String, String)
type Answer = [String]
oracle :: (Question -> Answer) -> Shake a -> Shake a
query :: Question -> Act Answer
The Generated File Problem

- Make has a problem with generated files (i.e. .hsc files)

```
$ cat Foo.c
#include <MyGeneratedFile.h>
```

- What are the dependencies of Foo.c?
  - Use the Make system to generate MyGeneratedFile.h
  - Read the contents of MyGeneratedFile.h for #include’s

- Faking it in Make
  - Run make twice (or more), first to generate files
  - Guess at the dependencies in advance
NO DEPENDENCY GRAPH!
History Traces

- A history trace is a list of question/answer pairs
  - What is the timestamp of Foo.c? 10am
  - What is the result of ls "*.c"? ["Foo.c"]
- When building a file, record the history
- Save that history to disk
- File is dirty if any answer has changed
- Alternatively: history is an abstract interpretation of a rule

```haskell
type History = [QA]
data QA = Oracle Question Answer
  | Need [(FilePath, ModTime)]
```
Implementation of need

```haskell
type Database = Map FilePath Status
data Status = Dirty History
             | Clean History ModTime
```

- Every Act accumulates a history
- In need:
  - Make sure the file is Clean
  - Add file/time to Act’s history
- If Dirty, rerun history, and if matching
  - Get file’s ModTime and switch to Clean
- If absent, or Dirty and history differs
  - If there is a matching rule, run it
  - If no rule but a real file, get it’s ModTime
Parallelisation

- need/want both take lists of files, which run in parallel

- Try and build N rules in parallel
  - Done using a pool of N threads and a work queue
  - need/want put their jobs in the queue
- Add a Building (MVar()) in DataBase

- Shake uses a random queue
  - Jobs are serviced at random, not in any fair order
  - link = disk bound, compile = CPU bound

- Shake is highly parallel (in theory and practice)
Profiling

- Can record every system command run, and produce:

  **Average was 3.42 (interquartiles were 4.00 and 4.00)**

**Hot Tools**
- 7.7s  $7 \times \text{gcc}$

**Hot Commands**
- 1.8s $\text{gcc -c file3.c -o file3.obj}$
- 1.8s $\text{gcc -c file2.c -o file2.obj}$
Future work: Shake --lint

- Parallel building often shows up build rule errors
  - In practice using a random queue makes these show up fast

- I want shake --lint, run once, in serial, guarantee parallel consistency
  - Can check the access times on all files
  - Check no files not in the history were accessed
  - Check all files in the history were accessed
Future work: also files

- Also files are annoying!
  - GHC builds .o files and .hi files in one command
  - Some things depend on the .o, some on the .hi
  - One rule modifies 2 database entries!

```
type Rule = FilePath
          -> Maybe (FilePath, Act ()
```

- Works, but impacts on lots of the core code
  - Not really a good model for also files
  - Potential for inconsistency
Practical Use

- Relied on by an international team of people every day
- Building more than a million lines of code in many languages

Before Shake
- Masses of really complex Makefiles, slow builds
- Answer to any build error was “make clean”

After Shake
- Robust and fast builds (at least x2 faster)
- Maintainable and extendable (at least x10 shorter)
- Creates a _database file to save the database
- Oracle is currently “untyped” (String’s only)
  - Although easy to add nicely typed wrappers over it
- Massive space leak (~ 12% productivity)
  - In practice doesn’t really matter, and should be easy to fix
- More dependency analysis tools would be nice
  - Changing which file will cause most rebuilding?
- What if the rules change?
  - Can depend on Makefile.hs, but too imprecise
- Not currently open source
Transitive Dependencies (theory)

- foo.c
  - dep: [a.h, b.h]
  - deps: [a.h, b.h, c.h, d.h]

- a.h
  - dep: []
  - deps: []

- b.h
  - dep: [c.h]
  - deps: [c.h, d.h]

- c.h
  - dep: [d.h]
  - deps: [d.h]

- d.h
  - dep: []
  - deps: []
Transitive Dependencies (Shake)

"*.c.dep" & "*.h.dep" *> \x -> do
src <- readFileLines $ dropExtension x
writeFileLines x
  [drop 8 s | s <- lines src, "#include" `isPrefixOf` s]

"*.deps" *> \x -> do
incs <- readFileLines $ replaceExtension x "dep"
let incs2 = map (<> "deps") incs
need incs2 -- parallel optimisation
writeFileLines x =<< concatMapM readFileLines incs2
Conclusions

- Haskell is a great language for a DSL
- A Make system is a DSL

- Any Make system based on a static dependency graph will fail to work with generated files
- Accurate dependency tracking is essential (i.e. Oracle)

- Shake is a Make system people actually like!