## Shake Before Building Replacing Make with Haskell

#### Neil Mitchell



community.haskell.org/~ndm/shake

### **General Purpose Build Systems**



- 🗵 Visual Studio
- 🗵 ghc –make
- 🗵 Cabal
- 🗹 make
- 🗹 Cmake
- 🗹 Scons
- 🗹 Waf
- 🗹 Ant
- ✓Shake

#### Generated files



 What headers does Foo.c import? (Many bad answers, exactly one good answer)

#### Dependencies in Shake



- Fairly direct
  - What about in make?

#### Make requires phases

Foo.o : Foo.c gcc -c Foo.o

Foo.o : \$(shell sed ... Foo.xml)

```
Foo.mk : Foo.c
   gcc -MM Foo.c > Foo.mk
#include Foo.mk
```

**Disclaimer**: make has hundreds of extensions, none of which form a consistent whole, but some can paper over a few cracks listed here

### **Dependency differences**

- Make
  - Specify all dependencies in advance
  - Generate static dependency graph
- Shake
  - Specify additional dependencies *after* using the results of previous dependencies



A build system with a static dependency graph is insufficient





# Shake at Standard Chartered



• In use for 3 years

– 1M+ build runs, 30K+ build objects,
 1M+ lines source, 1M+ lines generated

- Replaced 10,000 lines of Makefile with 1,000 lines of Shake scripts
  - Twice as fast to compile from scratch
  - Massively more robust

**Disclaimer**: I am employed by Standard Chartered Bank. This paper has been created in a personal capacity and Standard Chartered Bank does not accept liability for its content. Views expressed in this paper do not necessarily represent the views of Standard Chartered Bank.

### Shake vs make: 10x shorter, 2x faster\*

\*for one real example

#### Faster 1 of 4: Less work

gcc -MM finding headers has bad complexity
 At large enough scale, it really matters



Scan each header once, instead of once per inclusion

### Faster 2 of 4: Less rebuilds



commit decea285a863ff147f53d3748aac8b13
Author: Neil Mitchell <neil@bigproject.com>
Comment: MyGenerator, whitespace only

### Faster 3 of 4: More parallelism

- Big project ≈ perfect parallelism
  - No unnecessary dependencies
  - Depend on only part of a file
  - No phases (overly coarse dependencies)



### Faster 4 of 4: Better parallelism

• Random thread pool = 20% faster

- Avoid all compiles then all links



## Shake outside a bank

- At least 10 Haskell build libraries
  - 3 are Shake inspired implementations
- 2 Shake addon libraries

Zoom

Nemesis

Hake

Cake

Blueprint

There's a bit of scaffolding to get going, but the flexibility is really worth it to be able to handle auto-generated files easily.

#### More information

#### **ICFP** paper

#### Shake Before Building

Replacing Make with Haskell

Neil Mitchell ad and the 11 Synaid store

#### Also from eff.

Most complex sufficient projects a x complex using a hild tool  $y, \, y, \, \mathbf{x}, \, \mathbf{x}_{0}$  , which any commands in an other satisfying user define it de pardennies. Un fortunately, most hald tools manie a al-de parten nies to he specifie dive for the hald states. This matrix for makes many dependency pate as difficult to as pass, as painly door involving flow presented at hild time. We down how to effections this, matching, allowing additional dup naturalises to be specified while hilding. We have implemented on close in the Heddell library Shake, and have mad Shake to write a complex term. hild system which complex millions of lines of cole.

Categories and Schief Descriptors: D3 (Schoord) Pag antinine Language

General Terms Languages

Revent hild system, a on plation, Heikel 1. Introduction

A hild test, such as a to (Felderica 1978), takes a set of hild take, possioner input files, and penderes some on pit files. Using a to, a hild aloc on here incernes:

weak tay: Field Field

ar dreak arfiel field This min says that the file would be depends on the imputs filed

The strong time and the strong are presented to the import with and the d (in the large star and a second strong strong strong parent) line (). We assure that a software strong strong

not ely, the sector p and may system connectors prove this part on (Sectors have a mass con §2.5.) Using the build need we describe in this p -BOWERSHW RE

 $\begin{array}{c} \mathrm{d} \mathbf{b} \leftarrow \mathbf{\lambda} \rightarrow \mathbf{b} \\ \mathrm{out} \left[ \mathrm{d} \mathbf{a} \mathrm{d} \mathrm{d} \right] \\ \mathrm{d} \mathrm{d} \left[ \mathrm{d} \mathrm{d} \mathrm{d} \mathrm{d} \right] \end{array}$ 

contents +- walfielding "lists a"

and contacts  $a \sin^2 (a \sin^2 b)^2 = (a \sin^2 b)^2 + (a \sin^2 b)^2 = (a \sin^2 b)$ 

(C) goight notice will a ppe due e cour ' preprint' a pice is necessaril)

This was done blue hour to held consistency. We depend on This is do dow this how to had constitute. We depend on gravely do fills fixet at W and with live from fixet at its other wheth constants — being a fixed fits fluctuate should go into ou-self the. Note we depend on all the fluctuate strends, and finally will the two pregram. If where fixet at damps, every of the files fixed by first not damps, the result tar will be a build. The key difference of one subscitute should be don't hild tarkly. is that other than specifying all dependencies in advance, we allow in the rate pendencies to be specified agive examining the assoluof particus de perdensios. This difference is considito accustely describerary de perdensy relationship. Conside rite probem of dependencies stemming from the sinchilded by a C son a ciffle. So one build tools maps in these dependen-cies to be specified manually. Other thild tools allow two separate

process, where dependencies are computed to be the hild starts. Here if the hild system generates C files and then complex them, where the more by the grant much the mass much much with plot tables, the part is the discovery of the set a much fides.

#### L1 Contributions

We have in presented, on a full of and a car Hinde II library, named State, which is would affected in: . Since provides a non-in-long states for waiting its Million systems (5)) and my side all holps for formance in-formating in Million systems (5)) and my side all holps for formance in-formation (56). By importanting State was blocked it library as does not set to be with an assign for hill power of Handel it industry. the new of standards and from term to structure it age. In its systems. In addition to store a first the dependencies (§2), Shake also inchoice the important form as of as is, such as minimal whilds, (maning only a subset of the minimal score subset of the inprovides the get (523.2) and productions the field (meaning could ple independent mass of the same time, (543.2). We allow mass to opcome over range values, mot find not the files, all overing us to 1 as its conflict de pontonei es  $\{\frac{1}{2}, d\}$  and properly the state economies p ordering multiple out puts  $(\frac{1}{2}0,3)$ . We have their a mean for of useful tool sinte multiple cost parts (dec.). We show that is sum there of world work with the state of the state

 $^{2}$  is the M law long a is unit of 21 to eq.( ) are long a/a index

202222

#### 🔭 shake-0.2.5: Build system library, like Make, but more accurate dependencies. Development.Shake

This module is used for defining Shake build systems. As a simple example of a Shake build system, let us build the file result.tar from the files listed by result.txt:

contents <- readFileLines \$ replaceExtension out "txt"

import Development.Shake import Development.Shake.FilePath

need contents

#### main = shake shakeOptions \$ do want ["result.tar"] "\*.tar" $\gg \to -> do$

system' "tar" \$ ["-cf",out] ++ contents

Core of Shake Utility functions File rules Directory rules Additional rules Finite resources

Contents

|Contents |Index | Frames

We start by importing the modules defining both Shake and routines for manipulating FilePath values. We define main to call shake with the default shakeOptions. As the second argument to shake, we provide a set of rules. There are two common forms of rules, want to specify target files, and \*> to define a rule which builds a FilePattern. We use want to require that after the build completes the file result.tar should be ready.

Hackage (shake)

The \*.tar rule describes how to build files with the extension .tar, including result.tar. We readFileLines on result.txt, after changing the .tar extension to .txt. We read each line into the variable contents -- being a list of the files that should go into result.tar. Next, we depend (need) all the files in contents. If any of these files change, the rule will be repeated. Finally we call the tar program. If either result.txt changes, or any of the files listed by result.txt change, then result.tar will be rebuilt.