Overview of Language Support in Nix

Nicolas Mattia, Oct 25, NixCon 2019, Brno

1. The Basics
2. The Approaches
3. What now?

Enter The Matrix

- Fixed-output derivation
- Code generation (foo2nix)
- Lockfile based
- Pure-Nix

- UX
  - Just Works?
  - Consistent with user workflow?
- Incrementality
  - Changing a comment rebuilds everything?
- Evaluation
  - Shall I come back in 7 days?

Vocabulary

Compiler:	gcc	rustc	ghc	...
Build tool:	make	cargo	cabal	…

Nix ≥ Build tool > Compiler

2. The Approaches

Fixed-output Derivation

```
stdenv.mkDerivation
    {   name = "my-awesome-thing";
        src = <some/path>;
        buildPhase = ''
            doAbsolutelyAnything
        '';
        outputHashMode = "flat";
        outputHashAlgo = "sha256";
        outputHash = "1plh1qpj8y4n35g3yx936rzwhacjv7f9piafx4iar9i3azfywy11";
    }
```

Fixed-output Derivation

Verdict:

- Evaluation: FAST
- Incrementality: Bad
- UX: Bad

Can we improve?

- Only grab dependencies

Warning:

- Nix is *not* a CAS
- Gotta trust the buildPhase + remote content + compiler reproducibility
- Don’t forget the sha!

Code-generation (foo2nix)

```
$ bundix -l

<generates a gazillion
lines>
```

- Evaluation: depends
- Incrementality: good
  - (generally)
- UX: terrible
  - Manual steps
  - Huge commits

Can we improve the UX?

Lockfile-based

Self plug:

- github.com/nmattia/napalm
- github.com/nmattia/naersk

```
let
    naersk = callPackage sources.naersk {};
    napalm = callPackage sources.napalm {};
in
    [
        naersk.buildPackage ./rs {}
        napalm.buildPackage ./js {}
    ]
```

Lockfile-based

```
[[package]]
name = "aho-corasick"
version = "0.7.6"
source = "registry+https://github.com/rust-lang/crates.io-index"
dependencies = [
 "memchr 2.2.1 (registry+https://github.com/rust-lang/crates.io-index)",
]
...
[metadata]
"checksum aho-corasick 0.7.6
(registry+https://github.com/rust-lang/crates.io-index)" =
"58fb5e95d83b38284460a5fda7d6470aa0b8844d283a0b614b8535e880800d2d"
```

Equivalent:

```
mkDerivation {
    pname = "unpack-aho-corasick";
    version = "0.7.6";
    src = fetchurl {
        url = "https://crates.io/aho-corsaick/...";
        sha256 = …;
    };
    buildPhase = "tar -xvzf $src -C $out";
}
```

Lockfile-based

- Evaluation: bad
- Incrementality: ok
- UX: best in class

Pure-Nix

Build description

```
let
  lib =
    { src = ./src;
      dependencies = [ "wreq" "lens" ];
      extensions = [ "OverloadedStrings"];
    };
in
  { main = "Main";
    src = ./app;
    packages = [ lib ];
    dependencies = [ "wreq" "lens" ];
  }
```

Behind the scenes

```
let
    mkBuild = module:
        mkDerivation {
            name = "${module.name}-build";
            buildPhase = ''
                mkdir -p $out
                ghc $builtDeps/*.o ${module.src} -o $out/${module.name}.o
            '';
            builtDeps = module.moduleDeps;
        };
    topModule = ... ; # magic!
in
    doLink topModule
```

Pure-Nix

(snack et. al)

- Evaluation: TERRIBAD
  - But fixable?
- Incrementality: best in class
- UX: it's complicated
  - New tools to learn

3. What now?

RECAP

	UX	Incrementality	Evaluation
Fixed output	OUCH	OUCH	FAST
Fixed output (deps)	BAD	OK	FAST
Code-gen	BAD	N/A	N/A
Lockfile	GOOD	OK	SLOW
Pure-Nix	It’s complicated	BEST	BAD

Know Your Audience

newcomer

- Excellent UX
- Rest: meh

power-user

- UX can be bent
- Evaluation and incrementality must be good

(custom tools)

nixpkgs

- Eval must be FAST
- Rest: meh

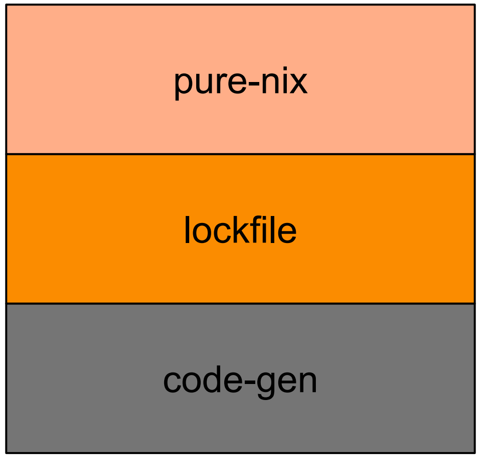
nix-hostage

- Perfect UX
- Fast eval
- Incremental

Take Away

- Incrementality:
  - requires build tool knowledge
  - constant fight with evaluator
- Hostages: need the best of all worlds
- Multi-tier?

Snack
(the ~~true~~ ideal story)



One more thing...

[...] any solution to the incremental build problem that involves the build system is going to be brittle and is going to involve essentially reimplementing the logic of your build system in Nix. The right way to go is recursive Nix.

`NixOS/nix/issues/13, @taktoa`

Is recursive Nix the new ccache?