NixOS? Umm, wat?

@NOIDD <RED@INFECT.ME>

Nix – Package Manager

Multi-platform:

- ANY Linux Distribution, (seriously - no more futzing with dep/pkg/rpm/et al.
- Any Architecture (does X86 / ARM out the box)
- OSX
- ChromeOS

Any OS. (some assembly required)

Feel free to port to ${OS_OF_CHOICE}

- We guarantee that we will not touch your underlying OS.
- We bring all our own tools, all our own dependencies.
- We literally don't care what OS we run under, period.

From BloatLinux to busybox, we don't care.

Nix is Magic

Nix is Functional

A functional definition of Functional:

- The same inputs ALWAYS give you the same outputs with no side effects (referential transparency)

Nix is Functional

A functional definition of Functional:

- The same inputs ALWAYS give you the same outputs with no side effects (referential transparency)
- The output is immutable.

Referential Transparency

Inputs:

- Source Code & Patches & Version (incl expression version)
- Configuration(ish)
- Any dependencies for your package
  - … and their code / patches / versions / dependencies.
- Time Travel for Deterministic Builds.

Referential Transparency

Outputs:

/nix/store/rj16lxwpsa9fn88lb45ds7yrw2jrc5rb-mtx-1.3.12/

./share
./share/man
./share/man/man1
./share/man/man1/loaderinfo.1.gz
./share/man/man1/tapeinfo.1.gz
./share/man/man1/mtx.1.gz
./share/man/man1/scsitape.1.gz
./share/man/man1/scsieject.1.gz

./sbin
./bin
./bin/tapeinfo
./bin/scsieject
./bin/scsitape
./bin/loaderinfo
./bin/mtx

Example Nix Package - mtx

```
{ stdenv, fetchurl }:
```

Example Nix Package - mtx

```
stdenv.mkDerivation rec {
  name = "mtx-1.3.12";
  src = fetchurl {
    url = "mirror://gentoo/distfiles/${name}.tar.gz";
    sha256 =
"0261c5e90b98b6138cd23dadecbc7bc6e2830235145ed2740290e1f35672d843”;
 };
```

Example Nix Package - mtx

```
meta = {
  description = "Media Changer Tools";
  longDescription = ''The mtx command controls blah blah blah...'';
  homepage = https://sourceforge.net/projects/mtx/;
  license = stdenv.lib.licenses.gpl2;
  maintainers = [ stdenv.lib.maintainers.redvers ];
  platforms = stdenv.lib.platforms.linux;
};
```

#include <ld.so> //wut?

Fscking Libraries how to they (normally) work?

#include <ld.so> //wut?

Fscking Libraries how to they (normally) work?

How does Nix bypass this?

#include <ld.so> //wut?

Fscking Libraries how to they (normally) work?

How does Nix bypass this?

What does it look like?

That's Pure!

… if only there was a OS that was built around this …

NixOS

Adds the concept of Declarative Configuration.

Imperative vs Declarative

Imperative, describes HOW to build the system.

Declarative describes what it is you want.

Hello NixOS

```
{config, pkgs, …}:
{
  imports = [
    ./hardware-configuration.nix
  ];
  networking.hostname = ”demo”;

… etc
```

evil.red – a guided tour

mc.evil.red – a guided tour

qmk_packaging


WAFFLE HOUSE