Nix Cook Book

tropf

ABSTRACT

My personal recipes when using nix ⟨`https://nixos.org/`⟩.

1. Intro

Nix refers to the operating system NixOS, the package manager and associated repository nixpkgs, and the functional language nix itself. I use all three, because I like its focus towards reproducability.

It declarative approach breaks with many habits and also tools, so in this document I attempt to make some notes how to tackle challenges I encountered in using nix.

2. Parameterized Packages in Flakes

To create multiple versions of the same package (derivation) with different parameters use `callPackage`. Note the trailing`{}` to immedeately create a v alid derivation.

```
mypkg = pkgs.callPackage ({doCheck ? false}: stdenv.mkDerivation rec {
  inherit doCheck;
  checkTarget = "test";

  cmakeFlags = (if doCheck then [
    "-DBUILD_TESTING=TRUE"
  ] else []);
}) {};

mypkg_with_tests = selfpkgs.mypkg.override {doCheck = true;};
```

3. Include nix CFLAGS in Exported Compile Commands

When exporting cmake compile commands the nix-defined compiler flags (stored in environment variables) will not be included. You can manually add them with a (hacky) sed command.

› This modifies JSON with sed and is evil. Use at your own risk.

```
nix develop && cd build
cmake -DCMAKE_EXPORT_COMPILE_COMMANDS=ON .
sed -e "s,$(which g++),$(which g++) $NIX_CFLAGS_COMPILE,g" -i compile_commands.json

# note: this is maybe already called automatically by your editor
rc -J .
```

4. Creating Debuggable Builds

By default, nix will enable all sorts of optimizations/security options ("fortifications") in C/C++ programs. This can make debugging very hard, as it optimizes code out/reorders your code, even using `Debug` cmake build type.

29 May 2023

To disable these modifications set: `hardeningDisable = [ "all" ]`
In full context this may be used as such:

```
buildInputs = [
    cmake
    gcc
] ++ (if doCheck then [
    catch2
    cmakeCurses
    gdb
] else []);

cmakeFlags = (if doCheck then [
    "-DBUILD_TESTING=TRUE"
    "-DCMAKE_BUILD_TYPE=Debug"
] else []);

hardeningDisable = if doCheck then [ "all" ] else [];
```

5. Build without Tests, but Develop with Tests

You can specify different targets for `nix build` and `nix develop` such that they refer to different deviations:

```
defaultPackage.x86_64-linux = mypkg;
devShells.x86_64-linux.default = mypkg_with_tests;
```

6. Ensure Locales Exist

The CI infrastructure I use lacks some locales. When building plots etc., the font handling doesn't work due to these missing locales. Disturbingly, my local `nix build` invocation finds the locales and looks completely fine, while the CI version does not. To fix this set the following variables inside the `stdenv` invocation:

```
LOCALE_ARCHIVE = "${glibcLocales.override {allLocales = false; locales=["en_US.UTF-
LC_ALL = "en_US.utf8";
```