Structure

László Kupcsik

July 20, 2024

Contents

1 Intro

2 Structure

2.1 Setup

2.1.1 Define the nix repo

2.2 List packages

1 Intro

`rix()` is the main function, so I'll start my exploration

2 Structure

2.1 Setup

The first part of the code just checks the various input parameters and emits warnings if needed.

It will also create a project directory, if needed. Then, it sets the path of `.Rprofile` and `default`.

2.1.1 Define the nix repo

Either bleeding edge, frozen edge or NixOS/nixpkgs. `make_nixpkgs_url()` takes care of this.

2.2 List packages

Package names are stored in `*_pkgs*` variables (character vectors). There's also a flag associated with each category. Empty `""` if it contains no packages, otherwsie a name, like `"rpkgs"`. This will be the name of the nix variable in a let binding, containing the package lists.

- CRAN: Current and archived CRAN packages
- TEX packages

- R packages from git
- Local R packages
- System packages

2.3 Generate default.nix

Using a set of `generate*` functions.

2.4 How R package names are parsed

The `get_rpkgs()` function splits the package vector into "pure" R package names, and versioned ones. Versioned names will always be stored in `archive_pkgs` and handled by `fetchzip`.

It also adds `{languageserver}` to the list of packages, whether it's there or not. Then, it collapses the packages into a `character(1)`, which is useful to pass into the nix expression.

The function returns a list of length 2, with `rPackages` (`character(1)`) and `archive_pkgs` (`character`).

Archived CRAN and git packages are handled by a single function. The archived CRAN packages come from the list above, and the git packages are defined by the user in a list. the `fetchpkgs()` function seems to handle both.

In reality, it just passes on the job to the `fetchzips()` and `fetchgits()`, which are wrapper functions for `fetchzip()` and `fetchgit()`. These will just create a nix expression using `buildRPackage`, which becomes an item in a nix list.

3 The plan

3.1 Collect inputs

3.2 Override rPackages tree

3.3 Create outputs