NixUP – declarative user profile

Thomas Strobel

28.10.2017

NixUP – a declarative user profile

- **central configuration file to specify:**
  - installed packages
  - running user services
  - managed files in home directory
- **tools to instantiate the user profile**

History of the idea

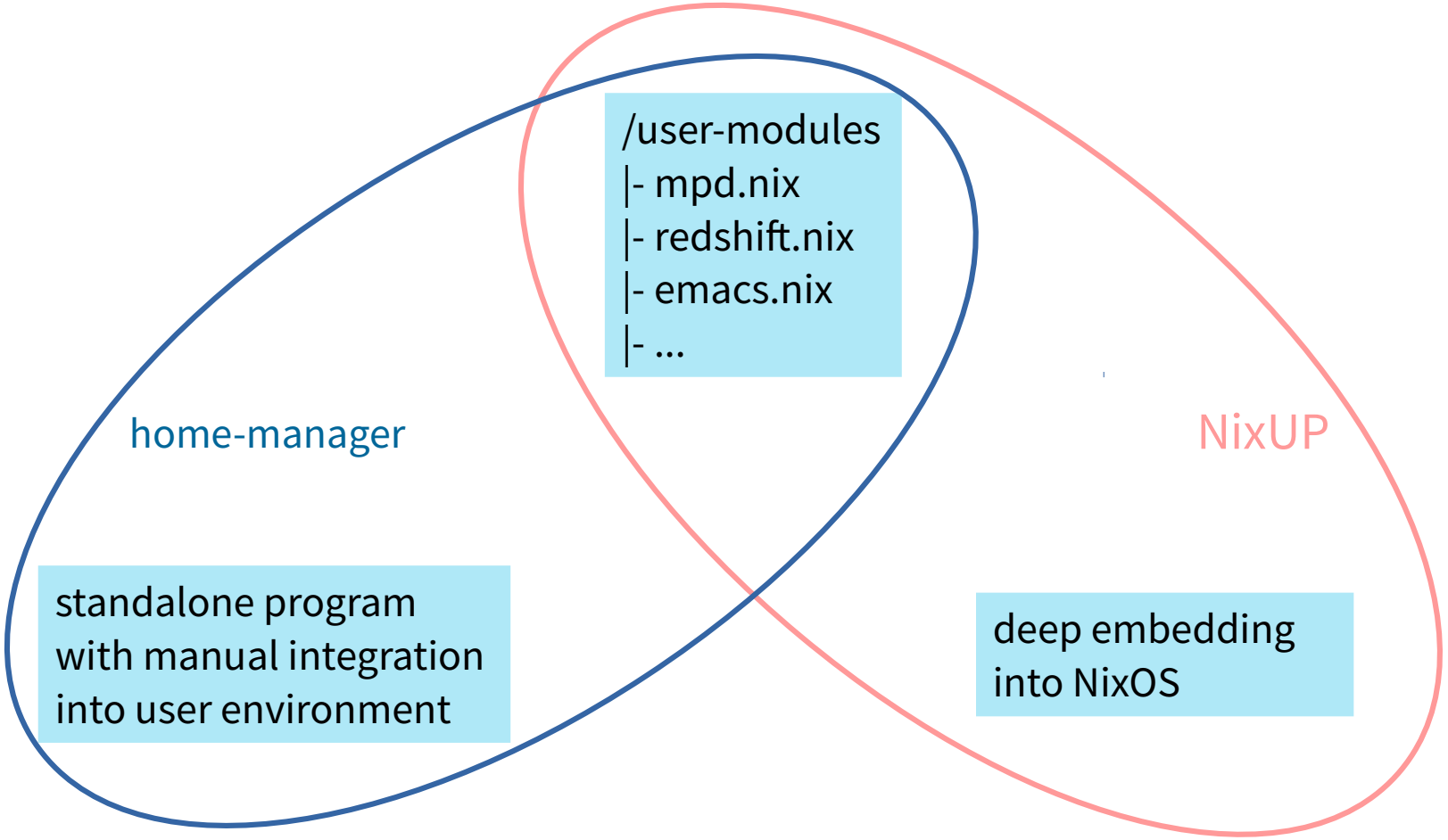
- first talking in Ljubljana in 2014
- first pull-request (*#4594*) by *Nicolas Pierron* in 2014
- long-standing PR (*#9250*) with 60 participants
- part of April Fools' joke in 2016 by *Peter Simons*
- *home-manager*: real implementation by *Robert Helgesson* since 2017

Worth the effort?

- problem: “likes” and “thumbs up” are cheap
- measure: asking for donations to the NixOS Foundation
- result: 665€ pledged from 21 donors – THANKS!

Design goal

- integrate with NixOS and home-manager



How to use NixUP

- enable NixUP in “/etc/configuration.nix”:

  ```
  config.nixup.enable = true;
  ```

- add declarative configuration in:

  ```
  .config/nixup/profile.nix
  ```

- activate nix user profile:

  ```
  nixup-rebuild switch
  ```

profile.nix

```
{ config, pkgs, ... }:
{
  user.packages = with pkgs; [ vim ];

  user.files = [
    { target = "nixcon2017.txt";
      content = ''
        Hi there! :)
      '';
    }
  ];

  systemd.services.mpd = {
    ...
  };
}
```

Internals of file management

- **key: managed files get extended filesystem attributes encoding:**
  - path of file relative to $HOME directory
  - initial content
- **extended attributes are checked before modification or deletion of file:**
  - existing files are protected
  - user-induced, manual changes are preserved

Features of file management

- sources can be:

source:	**static**	**dynamic**
content:	linked/copied	executed

- managed files can be:

file:	**non-volatile**	**volatile**
content:	protected	clobbered

Notes about file management

- atomic switching of managed files
- symbolic links target files or directories
- no atomic switching for a set of files
- order of execution during switch:
  1) stop deactivated user services
  2) switch managed resource files
  3) start new user services

Almost done!

- **help is needed with:**
  - testing
  - security review
  - writing documentation
  - merging into NixOS
  - merging with home-manager
- **donating to the NixOS Foundation as pledged :)**

THANKS!

Next project: Nix state management?

- **NixOS: purely functional Linux distribution**

  … for installing packages and services.

- **What about executing services?**
  - Guarantee a service has correct data to work on?

    e.g:

    1) nixos-rebuild: major update of a DB service
    2) systemd-prestart: upgrade DB to newer version
    3) nixos rollback: old service + new DB???

(type) effect system

- annotate required resources!

```
serviceA = {
  pkgs = pkgs.serviceA_36;
  working_dir = resource “ServiceA_36” ...;
};
```

- collect resource annotations during build
- check available resources before activation!