Nix для учёных

А. Худяков

19 июля 2018

Мотивация

Приезжайте! Мы научим вас собирать нашу библиотеку.

Мотивация

Приезжайте! Мы научим вас собирать нашу библиотеку.

Девиз

Человек должен думать — машина работать

- Сборка софта для пользователя должна быть тривиальна
- Рабочее окружение должно быть воспроизводимо
- Прошлогоднее рабочее окружение тоже должно быть воспроизводимо

Buzzwords!

Nix

- **Source-based пакетный менеджер**
  (можно использовать в любом дистрибутиве линукса!)
- **Декларативный**
  Пакеты описываются на чистом функциональном языке
- **Воспроизводимая сборка**
  Одинаковые никс-выражения дадут идентичные пакеты
- **Консистентность**
  Если изменяется пекет, все зависящие от него пакеты будут пересобраны
- **Герметичность**
  Все зависимости должны быть в никсе

Никс-выражения

Всё в никсе описывается при помощи языка никс-выражений.

Nix language

- Функциональный
- Чистый
- Ленивый
- Безтиповый

Никс-выражения

Всё в никсе описывается при помощи языка никс-выражений.

Nix language

- Функциональный
- Чистый
- Ленивый
- Безтиповый

```
$ nix-repl
Welcome to Nix version 1.11.16.

nix-repl> 2+2
4
```

Никс-выражения

Всё в никсе описывается при помощи языка никс-выражений.

Nix language

- Функциональный
- Чистый
- Ленивый
- Безтиповый

```
$ nix-repl
Welcome to Nix version 1.11.16.

nix-repl> 2+2
4

nix-repl> {a=1; b=3;}
{ a = 1; b = 3; }
```

Никс-выражения

Всё в никсе описывается при помощи языка никс-выражений.

Nix language

- Функциональный
- Чистый
- Ленивый
- Безтиповый

```
$ nix-repl
Welcome to Nix version 1.11.16.

nix-repl> 2+2
4

nix-repl> {a=1; b=3;}
{ a = 1; b = 3; }

nix-repl> {a = 1; b = assert false; 1;}
{ a = 1; b = "error:
  assertion failed at (string):1:17";
}
```

Пример никс выражения для сборки пакета:

```
{ stdenv, fetchhg, cmake, geant4npm, g4_support_cxx}:
stdenv.mkDerivation {
  name          = "mipt-npm-geant4-satellite";
  src = fetchhg {
    url = "ssh://hg@bitbucket.org/mipt-npm/geant4-satellite";
    rev = "0d499c7571be54342848762d71a896ff824c7773";
  };
  buildInputs = [cmake geant4npm g4_support_cxx ];
  propagatedBuildInputs = [
    geant4npm.passthru.data.G4ENSDFSTATE
    geant4npm.passthru.data.G4EMLOW
    ];
  cmakeFlags  = "";
}
```

Сборка пакета

Nix expression

↓

Nix derivation

↓

Собранный пакет

Комментарии

- **Derivation** — описание того, как надо собирать пакет
- **Всё** хранится в `/nix/store/`

  `/nix/store/03..1z-geant4-10.4.1.drv`

  `/nix/store/a9..gg-geant4-10.4.1`
- Пакеты становятся видимыми для пользователя если устанавливаются в его профиле.
- Коллекция всех пакетов никса — одна огромная программа.

Главные команды

nix-env

- Активирует и деактивирует программы в профиле пользователя.
- Собирает и устанавливает, если их нет

Главные команды

nix-env

- Активирует и деактивирует программы в профиле пользователя.
- Собирает и устанавливает, если их нет

nix-build

- Собирает и устанавливает пакеты в `/nix/store`

Главные команды

nix-env

- Активирует и деактивирует программы в профиле пользователя.
- Собирает и устанавливает, если их нет

nix-build

- Собирает и устанавливает пакеты в `/nix/store`

nix-shell

Наверное, самая интересная команда: создаёт интерактивную оболочку на основе никс-выражения.

Пример

shell.nix

```
let
  pkgs = import <nixpkgs> {};
  pyp  = pkgs.python36.withPackages (ps: with ps;
    [ jupyter_core notebook numpy
      scipy sympy matplotlib
    ]);
in pkgs.stdenv.mkDerivation {
     name        = "unfolding-shell";
     buildInputs = [pyp];
     };
```

run-notebook.sh

```
  nix-shell --command 'PYTHONPATH=$(pwd) jupyter notebook'
```

Заключение

Плюсы

- Решает проблемы, которые обещает решить:
  - Воспроизводимость
  - Простота сборки опакеченного
- Весьма удобен пока работает

Минусы

- Мало документации
- Отвратительный UX
- Малораспространён

Другое

- Аналогов не существует