

Дистрибуция на софтуер

Универсални конфигурации с Nix

Павел Атанасов Камен Младенов

01.04.2025

Раздел 1

Проблеми при дистрибуцията на софтуер

Стъпки при добавянето на нов софтуер

- Трябва да инсталираш програмата **и** зависимостите ѝ
- Инсталирането на програми/библиотеки се свежда до копиране на нужните файлове в някои **глобални** места: /lib, /bin, C:\Program Files, /Applications, ...
- Конфигурации се слагат на редица **глобални** (спрямо потребителя) локации: /etc, ~/.config, %APPDATA%, ...

- В модерни ОС (особено UNIX-базирани) се употребяват “пакетни мениджъри”
 - ▶ Примери: apt, yum, pacman, brew, ...
 - ▶ По-отдалечени примери: Apple App Store, Microsoft Store, Google Play Store, ...
- Даваш им редица уеб адреси в които да търсят програми и какво искаш да инсталираш.
 - ▶ Примери: <https://mirror.telepoint.bg/ubuntu/>,
<https://mirrors.uni-plovdiv.net/archlinux>, <https://mirrors.neterra.net/fedora>
- Те автоматично избират подходящите програми и зависимости за коректната архитектура/ОС/..., следят къде и какво се инсталира.

Ограничения

- ❶ Определянето на зависимости е трудно за поддържащите пакети хора
- ❷ Избирането на варианти за една програма рядко се поддържа
- ❸ Близко до невъзможно е да инсталираме повече от една версия на програма.

Възможно е две програми да ползват различни версии на зависимости!

- ❹ Обновяване може да презапише нещо неочаквано
- ❺ При премахването на програма трябва да премахнем зависимостите, които не са посочени от други програми.

Някои зависимости се появяват само по време на изпълнение.

Раздел 2

Как да разрешим тези проблеми?

Как да разрешим тези проблеми?

- Ако създаваме “чиста среда”, в която са налични само и единствено зависимостите, които сме определили, то ще сме сигурни в техния подбор
- Единственият истински начин да поддържахме всички възможни варианти е да позволим потребител да компилира програмата си
- Трябва да държим всяка програма и нейните стандартни данни (като конфигурации) в уникална директория, спрямо името, версията и варианта
- Обаче за проблемът със зависимостите...

Няма решение!

- Не е възможно да сме напълно сигурни дали програмата няма да генерира по неясен начин извикване към определена програма (без сериозен анализ на сорс-кода)
- Ще използваме някаква апроксимация, която би трябвало да работи в почти всички случаи

- Нека да четем резултатните *изпълними* файлове за референции към програми. Все пак пътищата във файловата система трябва да са текстови.
- В общия случай това не е достатъчно (факта че намираме думите `bin` и `firefox`, не гарантира че се ползва `/bin/firefox`)
- Трябва да **маркираме** нашите зависимости (по-точно директориите в които се намират) чрез някакъв уникален низ, който лесно се намира.

- Не искаме този низ (тоест името на директорията) да бъде напълно произволен, защото тогава бихме имали много копия на една и съща програма
- Не искаме и да зависи от резултатната програма, понеже два резултатни файла може да се различават по незначителен начин, въпреки че функционално са идентични
- Остава да зависи от *входните* данни за програмата - име, версия, вариант

Общо взето

- Искаме всяка програма, с всички допълнителни файлове, да се намира под уникална директория спрямо име, версия, вариант
- Трябва да можем да компилираме програми в среда с изчистени зависимости и при която всяка зависимост се реферира към наша уникална директория

- Компилирането е бавно и натоварващо
- Можем допълнително да чистим нашата среда, докато стигнем до **възпроизводим** (детерминистичен) процес
- Тоест, за дадени входни данни (програмен сорс, компилатор, настройки, ...), можем да сме абсолютно сигурни, че резултата няма функционално да се различава, колкото и пъти да пуснем процеса
- Така, можем да предоставим **кеш** - набор от компилирани пакети, които идентифицираме по входните им данни.

Ако целим да компилираме нещо, то търсим в кеша дали някой преди не го е компилирал, със същите входни данни. При съвпадение, използваме техния резултат.

Раздел 3

Nix пакетният мениджър

Nix пакетният мениджър

- Eelco Dolstra описва точно тези проблеми и решенията им преди >20 години
- Всяка програма се намира под `/nix/store/HASH-PROGRAM-VERSION`
- “Хешът” е редица символи, които се генерира от криптографски алгоритъм върху входните данни (включително настройките, определящи варианта).

Не е уникален, но на практика можем да приемем че е.

- За изчистената среда се използват специални възможности на Линукс ядрото и средата на изпълнение
- Ако програмата не е с отворен код (тоест идва като изпълним файл), използваме `patchELF`
- Изтриването на програма означава да обходим всяка зависимост и да проверим, дали се използва от някоя друга програма (чрез нашия апроксимиращ метод)

Инсталация

Linux и Windows (със Windows Subsystem for Linux, aka WSL)

```
sh <(curl -L https://nixos.org/nix/install) --daemon
```

MacOS

```
sh <(curl -L https://nixos.org/nix/install)
```


Раздел 4

Въпроси?