Functional Package Management using GNU Guix

Arun I

Indian Institute of Science, Bengaluru, India

November 20, 2019

Package managers are useful

- a convenient curated collection of software
- easy installing and updating packages
- automatic dependency management
- deduplication of dependencies for efficient disk and memory usage

Yet,

- systems break from time to time
- many niche packages may not be available
- version conflicts

Yet we are turning away from them

- application bundles (like AppImage) that snapshot an entire application along with dependencies
- containers (like Docker) that deploy snapshots of entire applications and services
- programming language or application specific package managers (like pip, gem, npm, etc.)
- motivated by cheap disk and memory and difficulty of packaging

Functional package management

- Functional (as in functional programming) package management
- Unique mapping from set of all dependencies and build inputs (say, the compiler) to the built output

$$\text{package output} = f(\text{package inputs}, \text{build inputs})$$

- Packages are built in an isolated build daemon where only the specified inputs are available

Guix profiles and the store

```
$ tree $(guix build mpop)

/gnu/store/4bl16fsnkbr2pkizh7msnngwxdqhib4z-mpop-1.4.5
|-- bin
|   |-- mpop

$ ls -d ~/.guix-profile

.guix-profile -> /var/guix/profiles/per-user/arun/guix-
    profile

$ ls -d /var/guix/profiles/per-user/arun/*

guix-profile -> guix-profile-271-link
guix-profile-269-link -> /gnu/store/67mfw6...-profile
guix-profile-270-link -> /gnu/store/yvkb21...-profile
guix-profile-271-link -> /gnu/store/2w9g8z...-profile
```

Guix profiles and the store

```
$ tree /gnu/store/yvkb21...-profile

|-- bin
|     |-- htop -> /gnu/store/lw83f5...-htop-2.2.0/bin/htop
|     |-- mpop -> /gnu/store/4bl16f...-mpop-1.4.5/bin/mpop

$ tree /gnu/store/2w9g8z...-profile

|-- bin
|     |-- htop -> /gnu/store/lw83f5...-htop-2.2.0/bin/htop
|     |-- mpop -> /gnu/store/4bl16f...-mpop-1.4.5/bin/mpop
|     |-- zip -> /gnu/store/kk0w04...-zip-3.0/bin/zip
```

Guix features

- perfect rollback of updates
- unprivileged per-user package management
- no version conflicts
- reproducible software environments
- powerful package customization system
- service configuration system

Perfect rollback

- Switching between profiles is an atomic process
- The previous profile contains all information about the previous system state without duplicating files on disk

Practical relevance

- No cold feet about upgrades; Upgrade anytime! Rollback if something breaks.

Unprivileged per-user package management

- Guix profiles enable many different versions of a package coexisting peacefully on a single machine
- Different user profiles can have different packages

Practical relevance

- Useful in shared HPC clusters where different users might need different packages, different versions of the same package, customized versions of the same package, etc.

Reproducible software environments

- Containers lack transparency; they are not easily inspectable
- Guix builds a reproducible environment from a text specification
- Possible to travel back in time (not just forward!) to previous versions of the operating system and packages

Practical relevance

- Reproducible science
- Software environments can be precisely controlled so that old computation can be reproduced exactly

Package definition and customization

- All packages are scheme objects
- Customization of packages is as simple as inheriting package objects and modifying their fields
- Complete Guile Scheme API for customization of all aspects of the operating system

Practical relevance

- Makes customization of the operating system as simple as writing a program

Example package definition

```
(define-public mpop
  (package
    (name "mpop")
    (version "1.4.5")
    (source
    (origin
    (method url-fetch)
    (uri (string-append "https://marlam.de/mpop/releases/"
                        "mpop-" version ".tar.xz"))
    (sha256
        (base32 "1m6743j8g777li..."))))
    (build-system gnu-build-system)
    (inputs
     `(("gnutls" ,gnutls)
       ("libidn" ,libidn)))
    (native-inputs
     `(("pkg-config" ,pkg-config)))
    (home-page "https://marlam.de/mpop")
    (synopsis "POP3 mail client")
    (description "mpop is a small and fast POP3 client
        suitable as a fetchmail replacement.")
    (license gpl3+)))
```

Example package customization

```
(package
  (inherit st)
  (arguments
   (substitute-keyword-arguments (package-arguments st)
     ((#:phases phases)
      `(modify-phases ,phases
         (add-after 'unpack 'configure
           (lambda _
             (substitute* "config.def.h"
               (("Liberation Mono:pixelsize=12")
                 "FreeMono:pixelsize=16"))
             #t))))))))
```

Operating system configuration and service deployment

```
(operating-system
  (host-name "steel")
  (timezone "Asia/Kolkata")
  (locale "ta_IN.utf8")
  (bootloader (bootloader-configuration
               (bootloader grub-bootloader)
               (target "/dev/sda")))
  (file-systems (cons (file-system
                        (device "rootfs")
                        (mount-point "/")
                        (type "ext4"))
                      %base-file-systems))
  (users %base-user-accounts)
  (packages (cons* curl htop nmap tree %base-packages))
  (services (cons* (service mongodb-service-type)
                   %base-services)))
```

References

- Largely based on Ricardo Wurmus' talk at FOSDEM 2017 https://archive.fosdem.org/2017/schedule/event/guixintroduction/
- Other talks by Guix maintainers https://git.savannah.gnu.org/cgit/guix/maintenance.git/tree/talks
- GNU Guix website https://guix.gnu.org
- GNU Guix reference manual https://guix.gnu.org/manual/