Practical, verifiable software freedom with GuixSD

David Thompson

Sunday, March 25th, 2018

about me

GNU Guix contributor since 2013

GNU Guile user and contributor since 2012

Day job: DevOps (AWS, Ruby)

the four freedoms

0: The freedom to run the program as you wish, for any purpose

the four freedoms

1: The freedom to study how the program works, and change it so it does your computing as you wish

the four freedoms

2: The freedom to redistribute copies so you can help your neighbor

the four freedoms

3: The freedom to distribute copies of your modified versions to others

the four freedoms

a wonderful set of rights, but often difficult to exercise in practice

common issues

figuring out how to view the exact source for a running program is tricky

common issues

building from source is difficult or sometimes impossible

- non-standard build system
- build scripts make assumptions that aren't true for your system

common issues

sharing source or binaries has many pitfalls

- dependency hell
- incompatible libraries between systems
- high barrier to entry for common package managers

common issues

major system upgrades can lead to sadness

ever upgrade your system, reboot, and find yourself in a completely broken state?

freedom: embedded

GuixSD removes many of the common barriers that prevent users from exercising their four freedoms

what is guixsd?

GuixSD is a fully-free GNU/Linux distribution with an advanced package manager and system upgrade mechanism

what is guix?

Guix is GuixSD's package manager (like apt, yum, pacman, etc.)

- unpriviliged package management
- per-user profiles
- atomic updates and rollbacks
- reproducible builds
- tools for many use-cases

unprivileged package management

users can build and install software without root privileges

tired: `sudo apt install emacs`

wired: `guix package -i emacs`

per-user profiles

each user may have one or more “profiles”, a union of many packages, without clobbering another user’s environment

use cases:

- Alyssa and Ben use different versions of Emacs
- Alyssa hacks on 2 Ruby projects that require different versions

transactional upgrades and rollbacks

experiment without fear!

```
guix package --upgrade emacs
```

oh no, the new version of Emacs is broken!

```
guix package --roll-back
```

transactional upgrades and rollbacks

system upgrades are transactional, too!

```
sudo guix system reconfigure my-machine.scm
```

oh no, the latest GuixSD updates broke my system!

no worries, just reboot and select the previous, working version from the bootloader menu

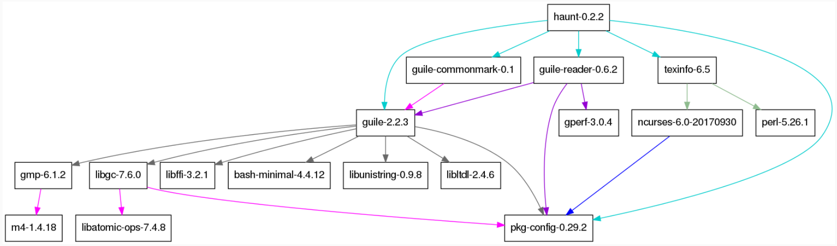
inspecting source code

quickly grab the source code for a package:

```
tar xf $(guix build --source gimp)
```

visualizing dependencies

```
guix graph haunt | dot -Tpng > graph.png
```



sharing packages

```
guix build -L ~/daves-packages foo
```

<minimal package example>

sharing development environments

```
(use-modules (guix profiles)
             (gnu packages base)
             (gnu packages guile))

(packages->manifest
 (list gnu-make
       guile-2.2
       guile-syntax-highlight
       haunt))
```

use it:

```
guix environment --manifest=guix.scm
```

experimenting in isolated environments

how about a container?

```
guix environment --container --ad-hoc ruby -- irb
```

sharing system configurations

```
(operating-system
  (host-name "izanagi")
  (timezone "America/New_York")
  (locale "en_US.UTF-8")
  (bootloader (grub-configuration (target "/dev/sda")))
  (file-systems (cons (file-system
                        (device "root")
                        (title 'label)
                        (mount-point "/")
                        (type "ext4"))
                      %base-file-systems))
  (users (list (user-account
                (name "dave")
                (comment "David Thompson")
                (group "users")
                (supplementary-groups '("wheel" "netdev" "audio" "video"
                                        "cdrom" "kvm" "input" "dialout")
                (home-directory "/home/dave"))))
  (packages (cons* arc-theme arc-icon-theme
                   htop less man-db ncurses nss-certs openssh unzip rsync
                   gnome-shell-extensions gnome-tweak-tool
                   %base-packages))
  (services (cons* (gnome-desktop-service)
                   %desktop-services))
  (name-service-switch %mdns-host-lookup-nss))
```

sharing binaries

start a server to share your builds:

```
guix publish
```

have a friend download them:

```
guix build \
     --substitute-urls=http://guix.example.com:8080 \
     hello
```

reproducible builds

reproducible builds produce bit-identical binaries when performed multiple times under the same conditions.

requires fixing issues in upstream build systems that are nondeterministic.

reproducible builds

this is a cross-distro effort, but Guix facilitates reproducibility more than others

see Chris Lamb's talk *"You think you're not a target? A tale of three developers. . ."* from yesterday for more perspective

reproducible builds

is this build reproducible on my machine?

```
guix build --rounds=3 hello
```

challenge authority

is this build reproducible on many machines?

is this build compromised?

`guix challenge`

customize packages

show me how Ruby is built:

```
export EDITOR=emacs
guix edit ruby
```

customize packages

let's make some changes!

```
git clone https://git.savannah.gnu.org/git/guix.git
cd guix
guix environment guix
./configure
make
guix build ruby
```

interoperate with other systems

need a Docker image?

```
guix pack --format=docker guile emacs geiser
```

(see *Solving the deployment crisis with GNU Guix* from LibrePlanet 2016 for reasons why Docker may not be so great)

interoperate with other systems

or maybe you want something like snap or flatpak?

make a tarball bundle that anyone can extract on their GNU/Linux system:

```
guix pack guile emacs geiser
```

interoperate with other systems

or maybe you want assistance translating foreign packages into Guix packages:

```
guix import pypi flask
```

literally: embedded

GuixSD now runs on the Beaglebone Black single-board computer!

```
(operating-system
  (bootloader (bootloader-configuration
               (bootloader u-boot-beaglebone-black-bootloader)
               (target "/dev/mmcblk1")))
  (initrd-modules (cons "omap_hsmmc" %base-initrd-modules))
  (services (cons* (dhcp-client-service)
                   (agetty-service
                    (agetty-configuration
                     (extra-options '("-L"))
                     (baud-rate "115200")
                     (term "vt100")
                     (tty "ttyO0")))
                   %base-services))
  ...)
```

extending guix

GuixSD is essentially a big Scheme library

with a little Scheme know-how its easy to write new tools that use the exact same APIs that the core Guix tools use

the freedom to contribute

GNU Guix is a welcoming community:

- we have a code of conduct and enforce it
- we have started seeking new contributors via Outreachy
- we participate in Google Summer of Code every year
- oh, and no copyright assignment (in case you were wondering)

join us!

thanks!

docs, past talks, source code, mailing list/IRC info, etc.:

https://gnu.org/s/guix

credits