GNU Guix: Package without a scheme!

Andreas Enge

andreas@enge.fr

GNU Hackers Meeting
Paris, 23 August 2013

Guix system

Two chunks of code

- Guix package manager
  - inside `guix/`
  - lots of Scheme
- GNU system
  - inside `gnu/`
  - Bootable system (`gnu/system/`), not yet written
  - Packages (`gnu/packages/`), almost no Scheme!

Available packages

- `guix package -A | wc`
  454
- 115 GNU packages (out of 364)
- GCC, 6 Schemes, Perl, Python, Ocaml
- TeX Live
- X.org (client side)
- Glib, Cairo, Pango, Gtk+
- Only free software!
- As pristine as possible (minimal patches)

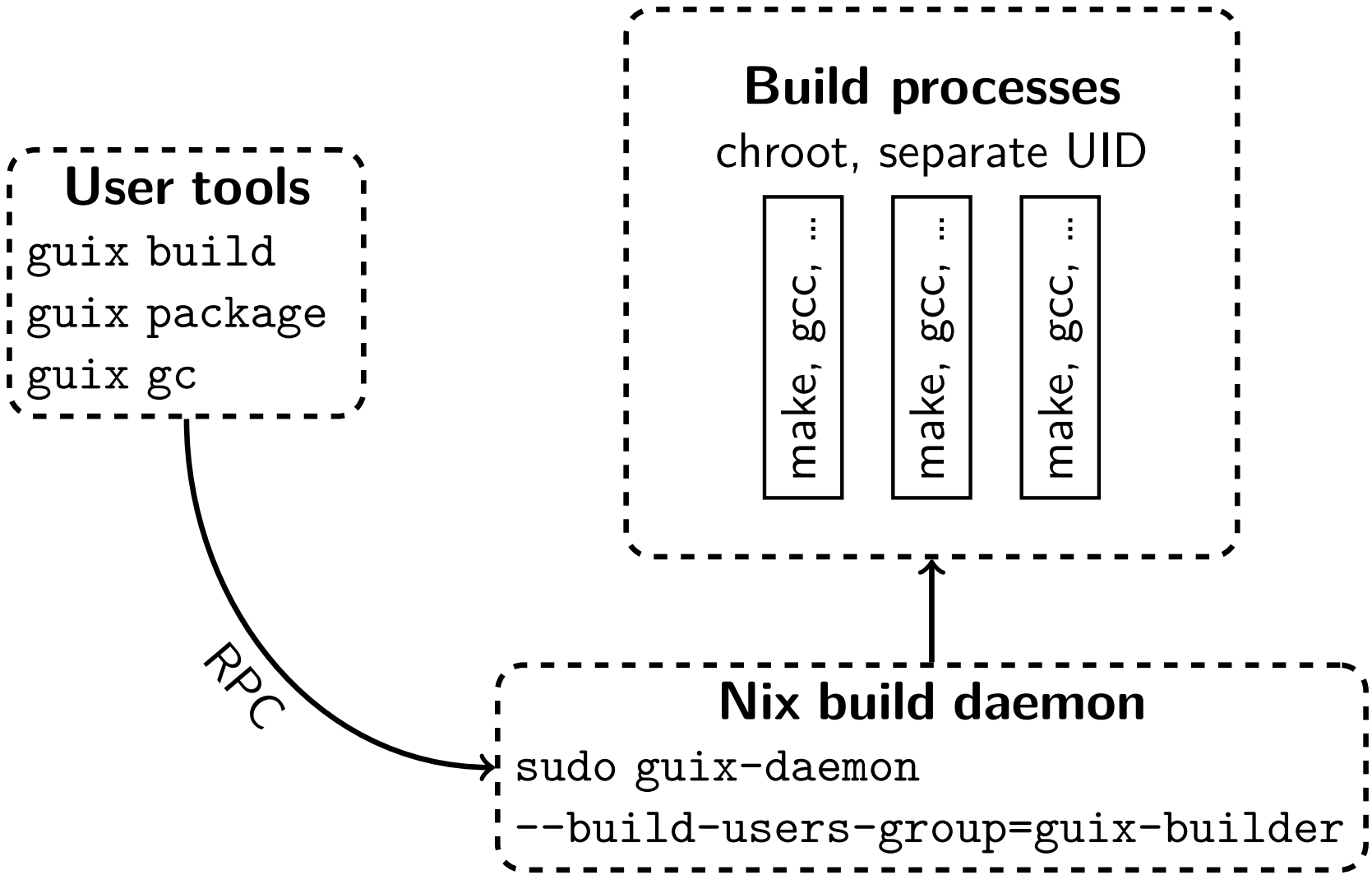
`http://www.gnu.org/software/guix/package-list.html`

Package description

- name: Lower case original package name
- version
- source
  - uri: Possibility of `mirror://xxx/...` with `xxx` $\in$ `gnu`, `sourceforge`, `savannah`, `cpan`, `kernel.org`, `gnome`, `apache`, `xorg`, `imagemagick`
  - sha256: Hash of tarball
  - build-system: `gnu-build-system`, `trivial-build-system`, `python-build-system`, `perl-build-system`, `cmake-build-system`
- synopsis: As in GNU Womb, not repeating the package name
- description: Usually taken from web page, two spaces at end of sentence
- license; see `guix/licenses.scm`
- home-page

Guix architecture

Courtesy of Ludovic Courtès



Let's scheme, at least a little bit!

- Scheme is functional: There are functions ...

```
2 + 3;                          (+ 2 3)
```

- ... and user-defined functions ...

```
int f (int x, int y)            (lambda (x y) (+ x y))
  return x+y;
f (2, 3);                       ((lambda (x y) (+ x y)) 2 3)
```

- ... but also global variables ...

```
a = 5;                          (define a 5)
2 * a;                          (* 2 a)
```

- ... that can hold functions

```
int f (int x, int y)            (define (f x y) (+ x y))
  return x+y;
f (2, 3);                       (f 2 3)
```

Let's list and quote

- Linked lists

```
(define a 1) (define b 2) (define c 3) (list a b c)
⇒ (1 2 3)
```

- Constant lists / quote

```
'(1 2 3)
⇒ (1 2 3)
(define a 1) (define b 2) (define c 3) '(a b c)
⇒ (a b c)
```

- Quasiquote

```
(define a 1) (define b 2) (define c 3) `(a b c)
⇒ (a b c)
```

- Unquote

```
(define a 1) (define b 2) (define c 3) `(,a b ,c)
⇒ (1 b 3)
```

Let's let

- Local variables

```
(define x 10)
(let ((x 2) (y 3)) (list x y))
⇒ (2 3)
x
⇒ 10
y
⇒ ERROR: Unbound variable: y
```

- Let's stop!
  - H. Abelson, G. Sussman, J. Sussman: Structure and Interpretation of Computer Programs, MIT Press, 2nd ed. 1996, `http://mitpress.mit.edu/sicp/`
  - D. Sitaram: Teach Yourself Scheme in Fixnum Days, `http://www.ccs.neu.edu/home/dorai/t-y-scheme/t-y-scheme.html`

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.
- Modify copyright, module name, name (twice), version, uri, description.
- Add module to `gnu-system.am`.

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.
- Modify copyright, module name, name (twice), version, uri, description.
- Add module to `gnu-system.am`.
- Download tarball:
  `guix download mirror://gnu/units/units-2.01.tar.gz`

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.
- Modify copyright, module name, name (twice), version, uri, description.
- Add module to `gnu-system.am`.
- Download tarball:
  `guix download mirror://gnu/units/units-2.01.tar.gz`
- Modify hash and license.

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.
- Modify copyright, module name, name (twice), version, uri, description.
- Add module to `gnu-system.am`.
- Download tarball:
  `guix download mirror://gnu/units/units-2.01.tar.gz`
- Modify hash and license.
- Check gpg signature.

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.
- Modify copyright, module name, name (twice), version, uri, description.
- Add module to `gnu-system.am`.
- Download tarball:
  `guix download mirror://gnu/units/units-2.01.tar.gz`
- Modify hash and license.
- Check gpg signature.
- Modify synopsis:
  `make sync-synoses`

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.
- Modify copyright, module name, name (twice), version, uri, description.
- Add module to `gnu-system.am`.
- Download tarball:
  `guix download mirror://gnu/units/units-2.01.tar.gz`
- Modify hash and license.
- Check gpg signature.
- Modify synopsis:
  `make sync-synoses`
- Build:
  `./pre-inst-env guix build units -K`

A simple GNU package: units

- Copy `gnu/packages/indent.scm` into a new source file.
- Modify copyright, module name, name (twice), version, uri, description.
- Add module to `gnu-system.am`.
- Download tarball:
  ```
  guix download mirror://gnu/units/units-2.01.tar.gz
  ```
- Modify hash and license.
- Check gpg signature.
- Modify synopsis:
  ```
  make sync-synoses
  ```
- Build:
  ```
  ./pre-inst-env guix build units -K
  ```
- Optionally: Cross-build
  ```
  ./pre-inst-env guix build units
  --target=mips64el-linux-gnuabi64
  ```

Update a GNU package

- Check for new version:
  `./pre-inst-env guix refresh units`
- Update package to new version:
  `./pre-inst-env guix refresh units -u`

A GNU package with inputs: pspp

- Copy-paste a package and modify as before, try to build.

A GNU package with inputs: pspp

- Copy-paste a package and modify as before, try to build.
- Add inputs, native-inputs, propagated-inputs.
- Add configure flags, try to build.

A GNU package with inputs: pspp

- Copy-paste a package and modify as before, try to build.
- Add inputs, native-inputs, propagated-inputs.
- Add configure flags, try to build.
- Work around problems. Here: Add missing input gettext. Try to build.

A GNU package with inputs: pspp

- Copy-paste a package and modify as before, try to build.
- Add inputs, native-inputs, propagated-inputs.
- Add configure flags, try to build.
- Work around problems. Here: Add missing input gettext. Try to build.
- Despair. Write a bug report.
  - Missing dependency on gnumeric?
  - Something related to libreoffice?
  - `--without-gui` not tested?
- Solution (thanks to John Darrington):
  `libxml2` was not found; add input `pkg-config`

A non-GNU package with build problems: gkrellm

- Dependencies according to `http://members.dslextreme.com/users/billw/gkrellm/gkrellm.html`: gtk, gdk, glib.
  Try build without inputs.

A non-GNU package with build problems: gkrellm

- Dependencies according to `http://members.dslextreme.com/users/billw/gkrellm/gkrellm.html`: gtk, gdk, glib.
  Try build without inputs.
- Following INSTALL, drop configure phase:

```
(arguments
`(#:phases
   (alist-delete
   'configure
   %standard-phases)))
```

  There is also `alist-replace`, `alist-cons-before`, `alist-cons-after`.

A non-GNU package with build problems: gkrellm

- We need to install into store, not `/usr/local`;
  and use `gcc` instead of `cc`.

```
#:make-flags
(let ((out (assoc-ref %outputs "out")))
  (list (string-append "INSTALLROOT=" out)
        "CC=gcc"))
```

A non-GNU package with build problems: gkrellm

- We need to install into store, not `/usr/local`;
  and use `gcc` instead of `cc`.

```
#:make-flags
(let ((out (assoc-ref %outputs "out")))
  (list (string-append "INSTALLROOT=" out)
        "CC=gcc"))
```

- Add inputs: gettext, gtk+, libsm, libice; native-inputs pkg-config

A non-GNU package with build problems: gkrellm

- We need to install into store, not `/usr/local`;
  and use `gcc` instead of `cc`.

```
#:make-flags
(let ((out (assoc-ref %outputs "out")))
  (list (string-append "INSTALLROOT=" out)
        "CC=gcc"))
```

- Add inputs: gettext, gtk+, libsm, libice; native-inputs pkg-config
- Missing `-lgmodule-2.0` at final link.
  Solution here: Additional make flag, see `src/Makefile`
  `X11_LIBS = -lX11 -lSM -lICE -lgmodule-2.0`

A non-GNU package with build problems: gkrellm

- We need to install into store, not `/usr/local`;
  and use `gcc` instead of `cc`.

```
#:make-flags
(let ((out (assoc-ref %outputs "out")))
  (list (string-append "INSTALLROOT=" out)
        "CC=gcc"))
```

- Add inputs: gettext, gtk+, libsm, libice; native-inputs pkg-config
- Missing `-lgmodule-2.0` at final link.
  Solution here: Additional make flag, see `src/Makefile`
  `X11_LIBS = -lX11 -lSM -lICE -lgmodule-2.0`
- No check phase:
  `#:tests? #f`
  Sometimes: `#:test-target "test"`

A non-GNU package with build problems: gkrellm

- We need to install into store, not `/usr/local`;
  and use `gcc` instead of `cc`.

```
#:make-flags
(let ((out (assoc-ref %outputs "out")))
  (list (string-append "INSTALLROOT=" out)
        "CC=gcc"))
```

- Add inputs: gettext, gtk+, libsm, libice; native-inputs pkg-config
- Missing `-lgmodule-2.0` at final link.
  Solution here: Additional make flag, see `src/Makefile`
  `X11_LIBS = -lX11 -lSM -lICE -lgmodule-2.0`
- No check phase:
  `#:tests? #f`
  Sometimes: `#:test-target "test"`
- Success!

Your turn!

- GNUnet
- Improvements to python build system
- Firefox
- GNOME, and other Gtk+ applications
- Qt (4.8 and 5.1)
- KDE
- OpenJDK
- LibreOffice
- Fonts
- Games
- Your favourite free package!

Subscribe to `guix-devel@gnu.org`.
Send your patches:

```
git format-patch
```

The source of this document is available from http://git.sv.gnu.org/cgit/guix/maintenance.git.