a gentle intro to functional package management with GNU Guix
Package managers are really useful.
Package managers are really useful.

But they can be so frustrating!
Version conflicts
Old software
Suspenseful upgrades
The only way is forward
Changes affect all users
Trust?
No package management
You are on your own.

External repos
EPEL, PPAs, AUR... more conflicts, more to trust

Application-specific packaging
gem, cabal, pip, cpan, npm:
more packages, less management

Build your own system package
Relocatable rpm, deb, PKGBUILD, ...

Meta package managers
Generate system packages with e.g. fpm

Giving up
Packaging is hard, let’s take snapshots.
headers
sources
build tools
libraries
...

\[ f(x) \]

cabba9e-\textbf{emacs-24.5/}
├── bin
│   └── Emacs
└── lib
    └── ...
(define glpk
  (package
    (name "glpk")
    (version "4.57")
    (source
      (origin
        (method url-fetch)
        (uri (string-append "mirror://gnu/glpk/glpk-
                        version ".tar.gz"))
        (sha256 (base32 "0p17jj1ixd2...")))
      (build-system gnu-build-system)
    (inputs
      `(("gmp" ,gmp)))
    (arguments
      `(^:configure-flags "--with-gmp")
      (home-page "http://www.gnu.org/software/glpk/")
      (synopsis "GNU Linear Programming Kit")
      (description
        "GLPK is a C library for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems ...")
      (license gpl3+)))
GNU Guix

Guix client
(guix packages)
(guix store)

RPCs

Build daemon

Isolated build processes
chroot with declared inputs

Store
Learn more!

#guix on irc.freenode.net
http://gnu.org/s/guix

rekado@elephly.net