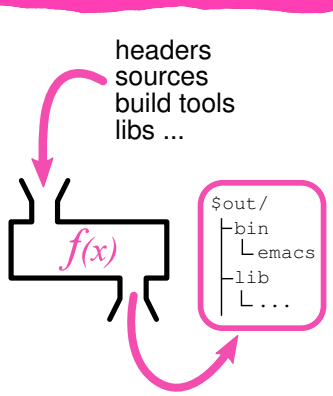
GNU Guix

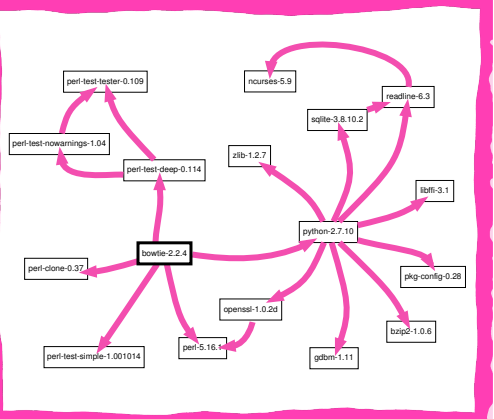
Functional package management at the core of the GNU system

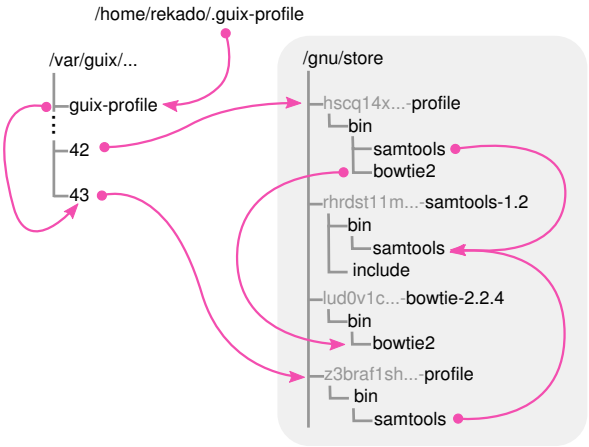
Functional package management

The package build is seen as a **function** in the mathematical sense, taking **inputs** (build scripts, compiler, libraries, sources), and **returning an installed package**.

As a **pure function** its result depends **only** on its inputs; there is **no global state**. Just like the result of a pure function can be cached, the package output directory is cached in the **store**.







Independent software profiles

The name of each output directory is **unique** as it is derived from **all** inputs. A **software environment** can be built by creating the **union** of the output directories of all desired packages.

These software **profiles** can be **independently** managed **by users** with Guix. As a profile is just a forest of symbolic links to immutable items in a shared store, users can **roll-back** to previous versions of the environment any time, and to install **different variants** of applications and libraries using **separate** profiles.

GNU inside!

100% **free software**, available through a unified interface, curated by GNU hackers. Guix is designed for practical software freedom.

Reproducible

Share exact replicas of software environments without the need for clunky disk images. Or play with software in an ad-hoc environment.

```
(list
  guile
  guix
  :-)
#t
```

Hackable and liberating

All packages are just **Guile Scheme** values, so they can easily be modified without expert knowledge. In fact, the **whole operating system** configuration is declared with a clean Scheme DSL.
Happy hacking!

Dependable

There is no need to worry about breaking your system as you experiment with your practical software freedom: you can always roll back to previous versions.

http://gnu.org/s/guix