Reproducible bioinformatics software environments with GNU Guix

System Admins prefer mature software, no variants, only apply unavoidable updates

Users want fresh software, multiple variants, latest tools, flexibility

Compromise
Use stable software for systems; let users manage their software stack on their own. This often leads to environments with following properties:

- There is no way to reproduce the environment, even on the same machine at a different point in time.
- Due to a lack of isolation, the environment will change or break when the host system changes.
- No safe upgrades or roll-backs. No separation for different workflows. Unportable.

ad-hoc
volatile
primitive

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 solely 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.

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. A profile is just a forest of symbolic links to items in the shared store. This enables a user to roll-back to previous versions of the environment, and to install different variants of applications and libraries using separate profiles.

Towards reproducible research

As each package captures the complete dependency graph, down to the kernel headers, all dependencies are immutable and are satisfied by the store. There are no dependencies on the host system, enabling the sharing of well-defined software environments across machines, aiding reproducible research.


Ricardo Wurmus (AG Akalin) ricardo.wurmus@mdc-berlin.de