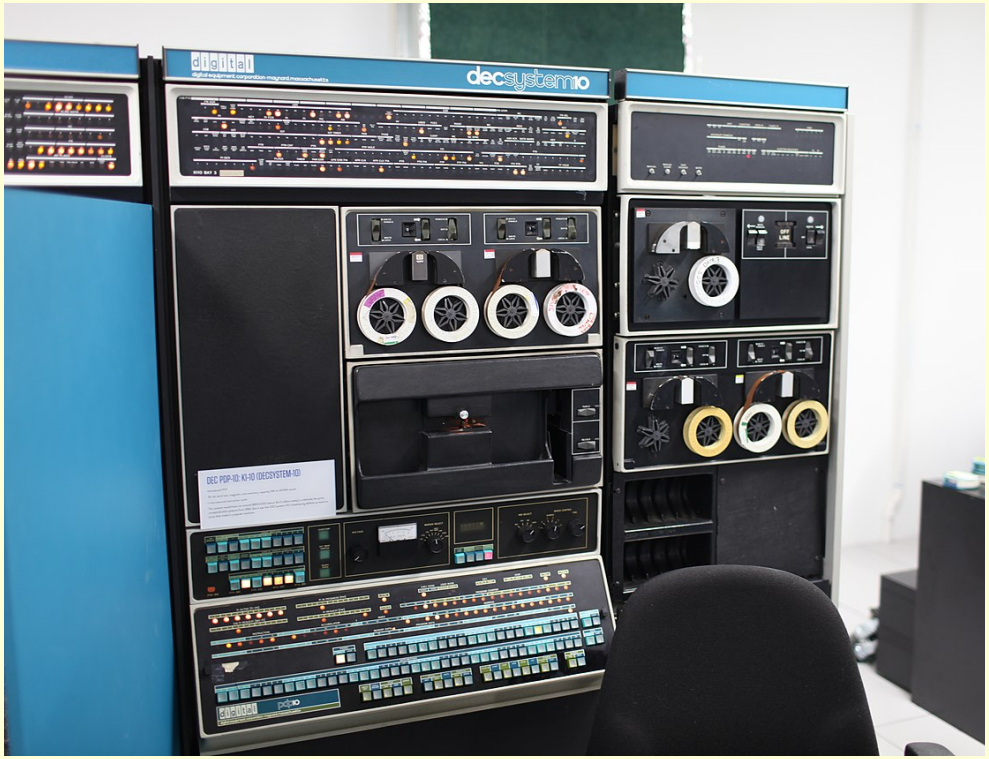

Distributing Freedom
How package managers empower software users

Before package managers…

- Most computer systems were time-sharing mainframes or minicomputers
- A ‘sysadmin’ (system administrator) would manually install software from tapes



DEC PDP-10

Before package managers...

- Software was distributed as a tar – ‘Tape Archive’ – of source code
- The source files were extracted, compiled and linked together on the machine
- On UNIX and Linux operating systems, the compiled files were put in /usr/local/

The first package managers

- Personal computers had no system administrator!
- Package managers allowed PC users to install, update and remove software easily

IBM 5150

The first package managers

1993: **Bogus Linux** pms

1994: **FreeBSD** pkg_* suite

1994: **Debian** dpkg

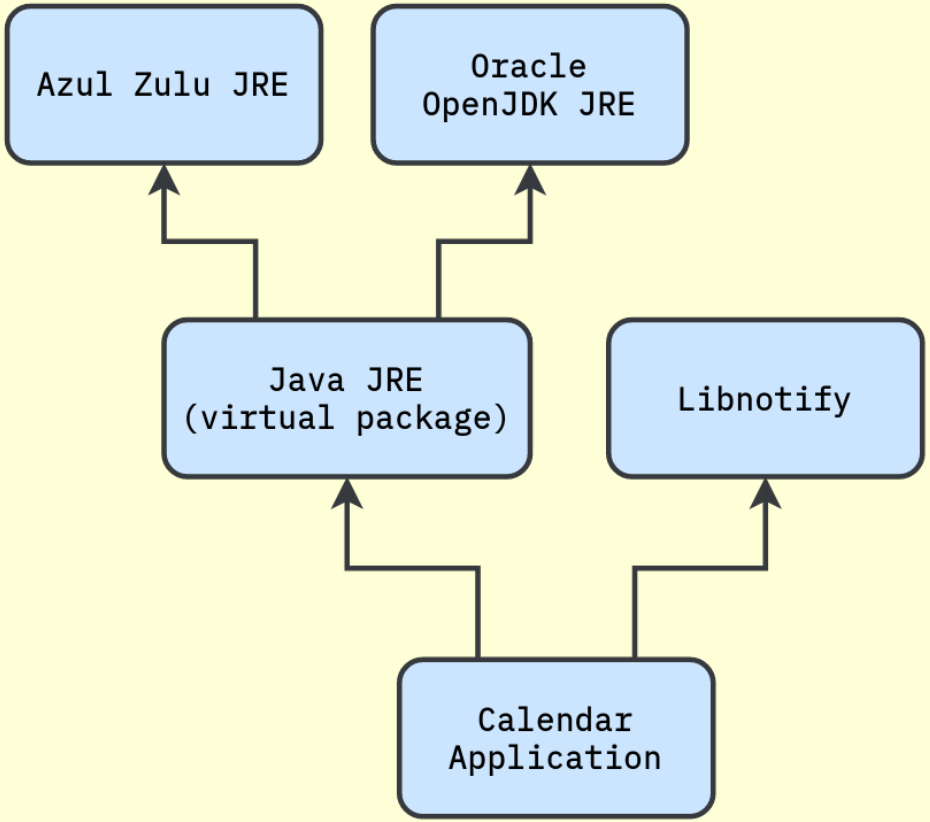
1995: **Red Hat** RPM

1999: **Gentoo** Portage

Fundamental features

- Installing software
- Removing software
- Listing information about installed packages
- Searching for and downloading software from the internet
- Automatic dependency resolution...

Automatic dependency resolution



- Applications usually need libraries to run
- Virtual packages allow choosing among multiple compatible versions

Package managers post-millennium

- Gradual improvements to existing package managers
- Alpine Linux and Arch Linux create package managers similar to those of FreeBSD and NetBSD
- Package managers come to mobile devices

Nokia N9

Proprietary package managers

2008: **Apple App Store**

2008: **Android Market**

2009: **Ninite** for Windows

2011: **Windows Store**

2012: **Google Play**

Proprietary package managers

- App stores encouraged developers to ‘give away’ their software, or sell it very cheaply
- To keep profits up, further disruptive functionality emerged in proprietary software
  - Pay-to-win games
  - Adverts in applications
  - Privacy-limiting analytics

Proprietary package managers

Most proprietary app stores are technically limited!

What package managers **could** provide users:

- Stability
- Free licensing
- Security
- Ease of use

User empowerment: Stability

- Automatic dependency resolution gets the *right* packages, and uninstalls packages that are no longer needed
- Users can easily remove faulty packages
- Allows the user to ‘roll back’ to a previous working version

User empowerment: Free licensing

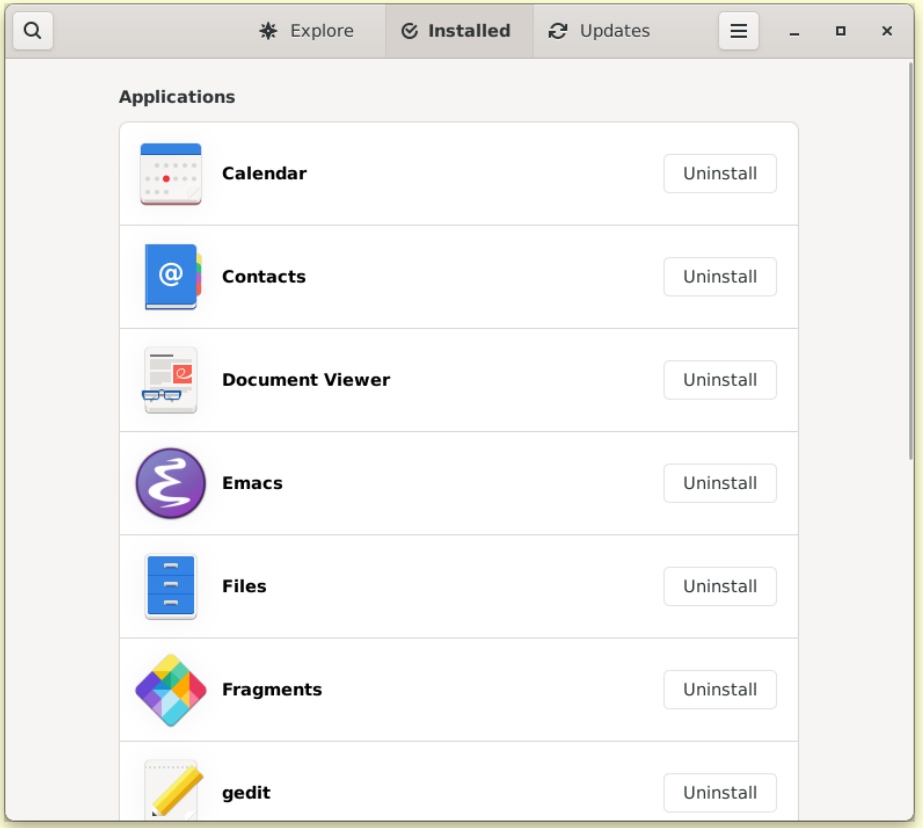
- Package managers display information about the software's licence
- Distributions' repositories have licensing policies
  - Debian Free Software Guidelines
  - Fedora 'Good' List
- SPDX License Identifiers reduce complexity of different licences

User empowerment: Security

- Cryptographic signing ensures that packages are authentic and undamaged
- Package-level signatures alleviate the need for HTTPS; HTTPS can still be used for anonymity
- Software with known vulnerabilities can be updated automatically

User empowerment: Ease of use

- Users can discover software from large repositories
- Graphical user interfaces make common usage simpler



GNOME Software

The future of package managers

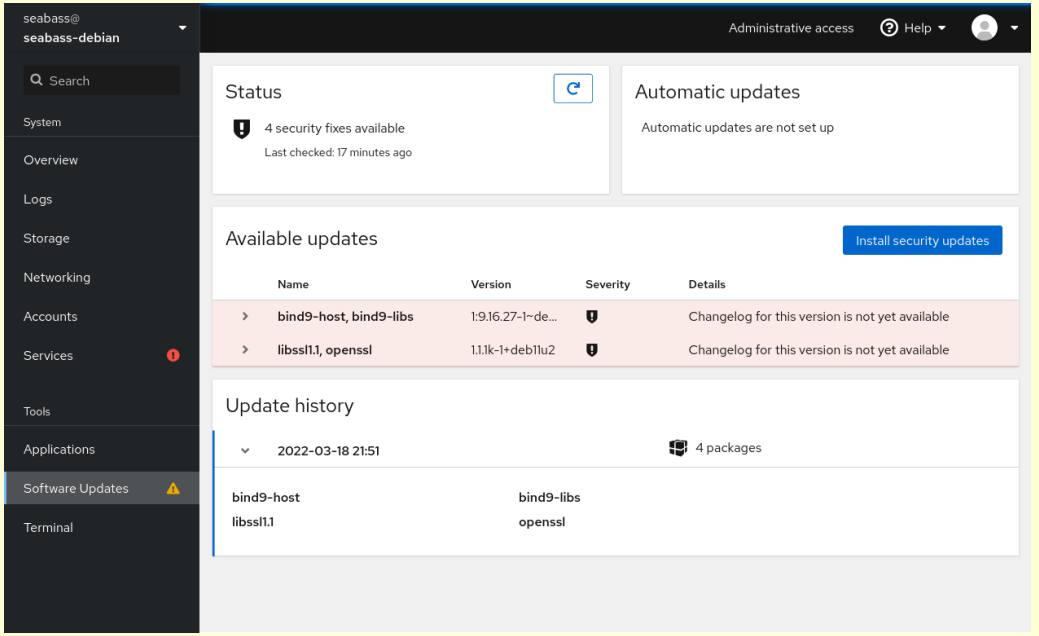
Package management has so much left to be explored!

There is active development in the fields of:

- Remote control across devices
- Application containerisation
- Declarative package management
- Supply chain security

The future: Remote control

- Remote access for all devices
  - Smart televisions
  - Mobile devices
  - Home appliances
- One place to securely update all devices



Cockpit web interface

The future: Application containers

- Containers allow the separation of programs
  - Sandboxing keeps users' data private from snooping software
  - Allows the user to run usually conflicting software simultaneously
- Current technologies include Flatpak and OCI (Open Container Initiative) runtimes

The future: Declarative management

- The packages for a system are declared together, rather than being installed one by one
- Allows easy migration between physical machines or installation across many machines
- Nix, GNU Guix and Fedora Silverblue are leading examples

The future: Supply chain security

- Deep dependency trees can conceal vulnerable code, allowing problematic packages to spread
- Automated security audits can be combined with Software Bill of Materials (SBOM) to alert users of potential problems
- Provenance data and Attestations ensure that packages were built securely

Image credits

- Title picture – Nathan O'Nions, CC-BY-2.0
- PDP-10 at the Seattle Living Computer Museum – Jason Scott, CC-BY-2.0
- IBM Model 5150 – Rama & Musée Bolo, CC-BY-SA-2.0 (France)
- Nokia N9 – Animist, CC-BY-SA-3.0
- GNOME Software – GPL-2.0-or-later
- Closing picture – Solomon203, CC-BY-SA-3.0 (modified from the original)
- LibrePlanet 2022 Logo – Free Software Foundation, CC-BY-SA-4.0


LIBREPLANET 2022
LIVING LIBERATION
Thank you for listening!
Sebastian Crane