The Haiku Package Manager

Richard Zak
richard.j.zak@gmail.com

10 November 2021

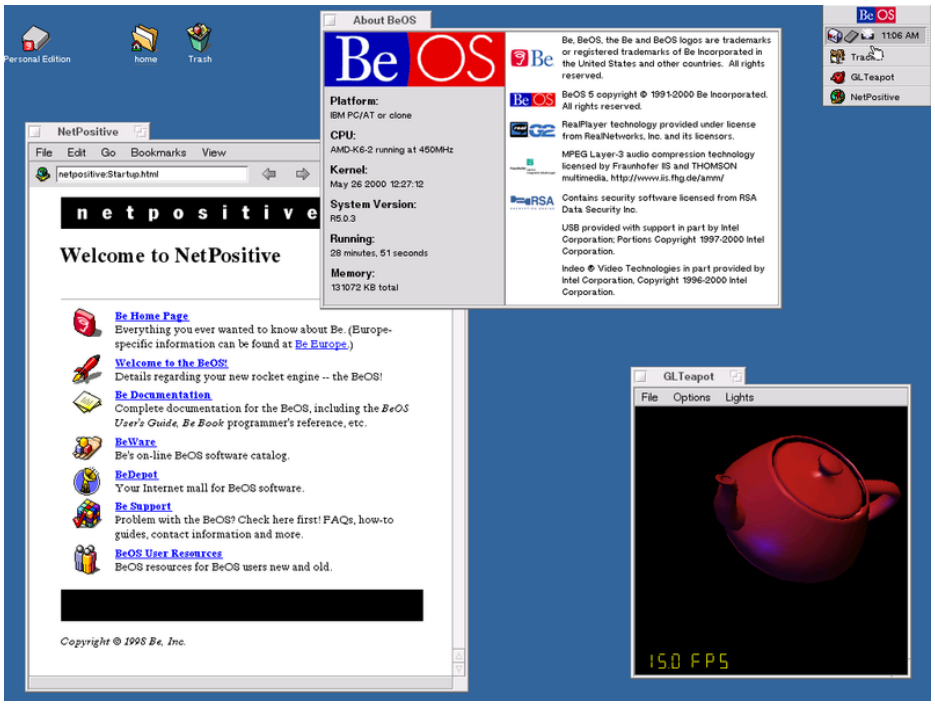
whoami

- My name is Richard Zak
- Haiku enthusiast, software developer, cybersecurity researcher
- I'm not a member of the Haiku team
- Tinkering with computers since the mid-90's
- `https://rjzak.github.io/`
- richard.j.zak@gmail.com

Haiku: History

From Haiku's website:

> *Haiku is an open-source operating system that specifically targets personal computing. Inspired by the BeOS, Haiku is fast, simple to use, easy to learn and yet very powerful.*



BeOS image from ArsTechnica.com.

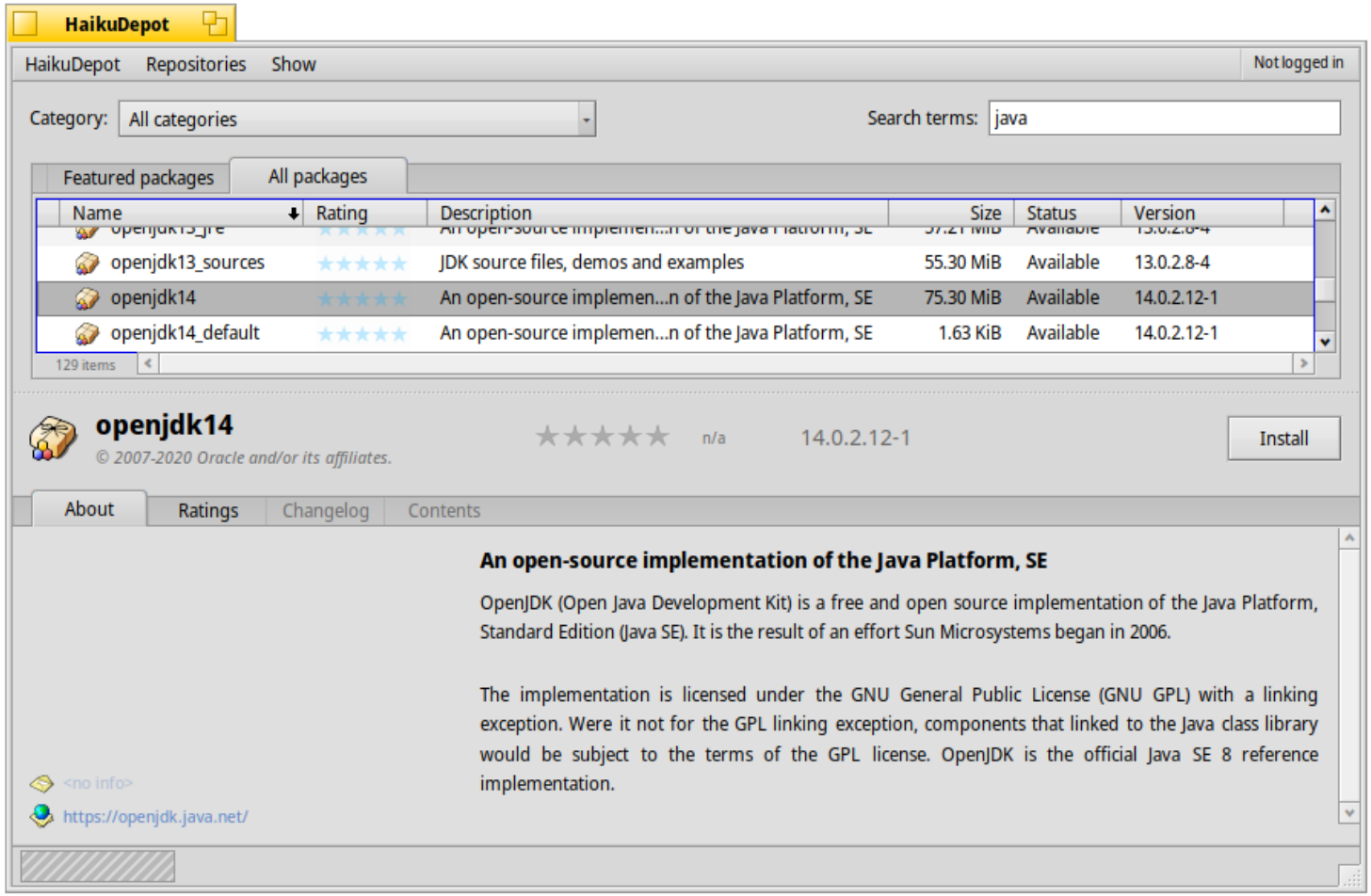
What is Haiku?

- Single-user, Unix-inspired, desktop operating system
  - Executables and libraries are ELFs
  - Bash is the default shell for the Terminal
  - SSH is included, client & server
  - Many programs work on Haiku with a simple recompile, simple build changes, simple header changes.
  - Most programs are compiled with Make, CMake, gcc. LLVM is available too.
  - It's not Linux, BSD, etc. Haiku has it's own unique kernel.
- Inspired by BeOS, which was inspired by the Classic Mac OS
- Exists solely because of it's developer & user community, volunteer efforts, and donations. There's no corporate sponsorship.
- Now works on RISC-V. Work is in progress for ARM64 support, plus others.
- ... and it's 20 years old! The project was formed in 2001, with it's first release in 2002.

Haiku Packages

The usual features:

- Separate packages for the application, development, source
- Retrieval of packages, including dependencies, from a repository
- Architecture-aware



Haiku Packages

The usual features:

- Separate packages for the application, development, source
- Retrieval of packages, including dependencies, from a repository
- Architecture-aware

Some unique features

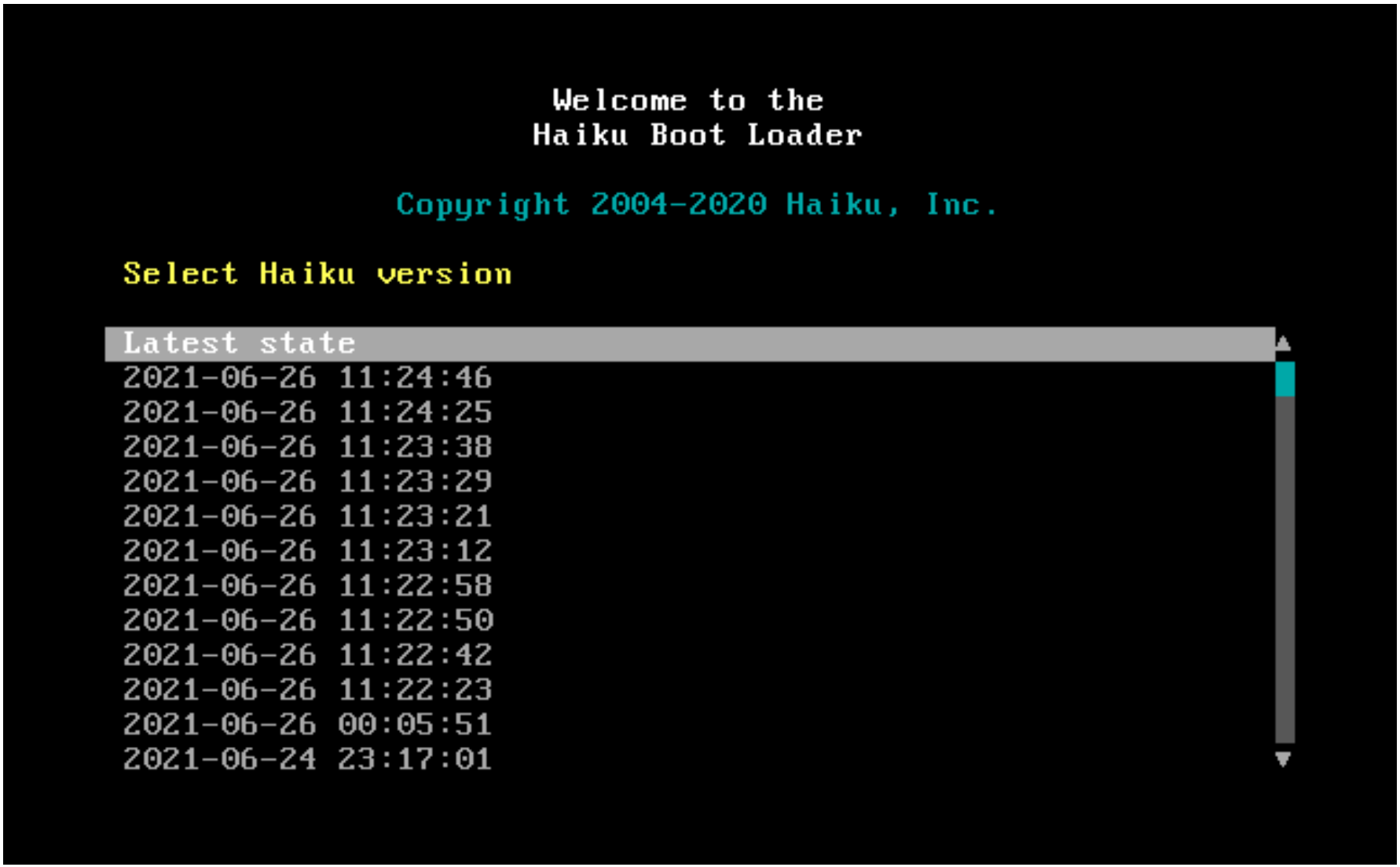
- Package contents are mounted as read-only into the file system
- Packages can indicate which libraries and/or commands are provided, allowing for easier dependency management
  - Packages can have a dependency specified as `lib:jpeg` vs. `libjpeg-2.0.3`. Because it is `libjpeg` or `libjpeg-turbo8`, for example?
- The file system knows which package is the origin for a given file.
- The boot loader knows about the package manager.

Packaged Files & Directories are Read-Only

Since the package contents are mounted from the package itself on to the file system, the files and directories are read-only.

- Made possible by separating the package file into equal-sized chunks and compressing individually, enabling cheap random access into the compressed file.
- This benefits the user, since it's guaranteed that the file can't be modified, either accidentally or intentionally.
- This benefits the developer, since it's guaranteed that the program and it's supporting files will be present in the known relative paths.
- The user can side-step this, if desired, to install software in `/boot/system/non-packaged/`
- Since the installed packages are on disk, making them redistributable (if offline, for example) is possible.

The Boot Manager is Package-Aware



The Boot Manager is Package-Aware

- Prior states allowing going back to how the system was previously
  - Including Haiku itself, since it's also in a package.
- Immensely helpful for operating system development, debugging
- Prevents the user from being completely locked out of their system

Metadata in Packages

Package information:

- Basics: name, description, software license
- Provides: indicates if a package provides a library, `lib:foo` or command, `cmd:bar`
- Other packages, and the command line, can specify dependencies on by referring to the the library name or command without having to specify which package exactly.
- Example: `pkgman install cmd:foo`

Considerations when Porting

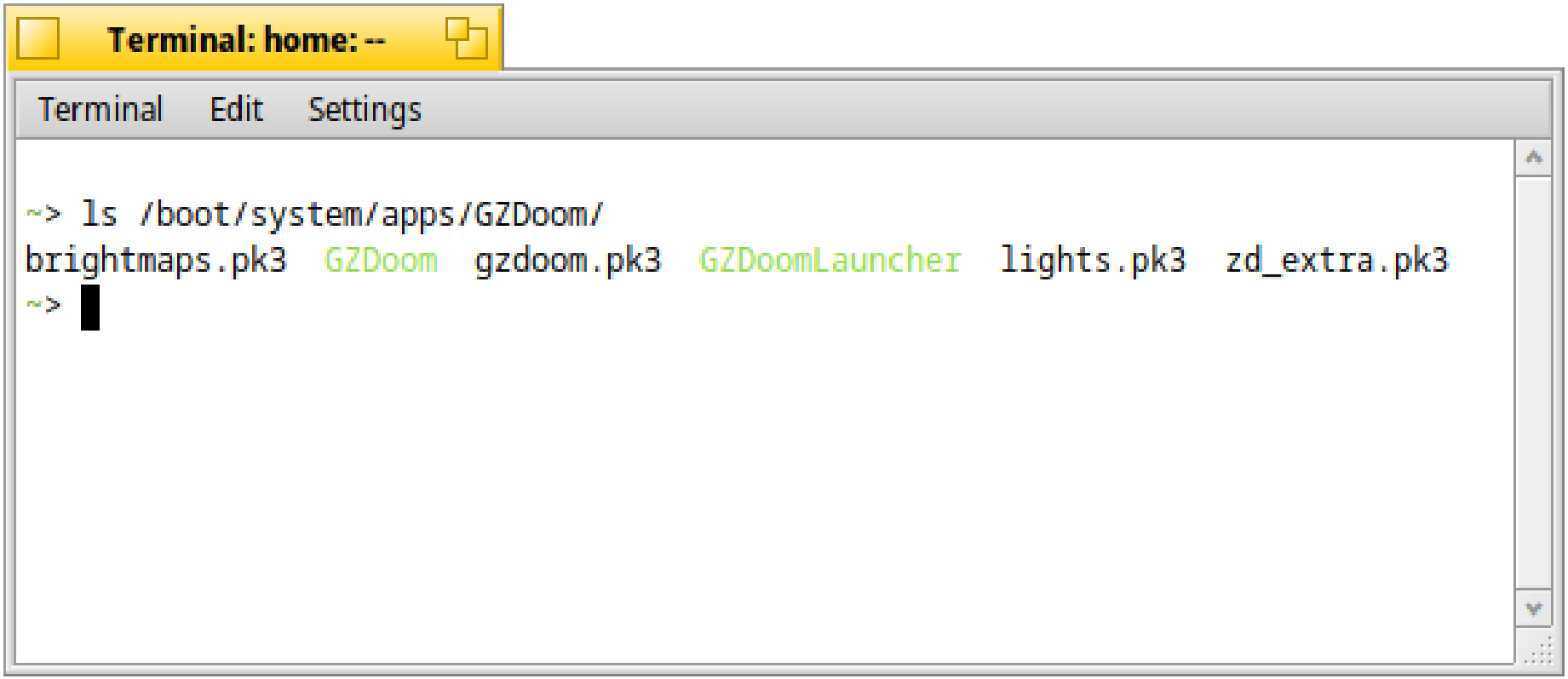
Generally, from a packaging perspective, porting to Haiku isn't too difficult. But there are some things to be considered:

- A lot of the same files exist on Haiku, but in different locations.
  - `/etc/resolv.conf` → `/boot/system/settings/network/resolv.conf`
  - Configuration dot files in the home directory *should* now be in `/boot/home/config/settings/` (convention)
- Directories from a package are read-only, so, for example, using Python's `pip` becomes problematic, as `/boot/system/lib/python3.7/` and it's subdirectories are read-only. Solution:
  - Package the desired module into a Haiku package, or
  - install the package in `/boot/system/non-packaged/lib/python3.7`, or
  - create a virtual environment in the home directory, `/boot/home`.

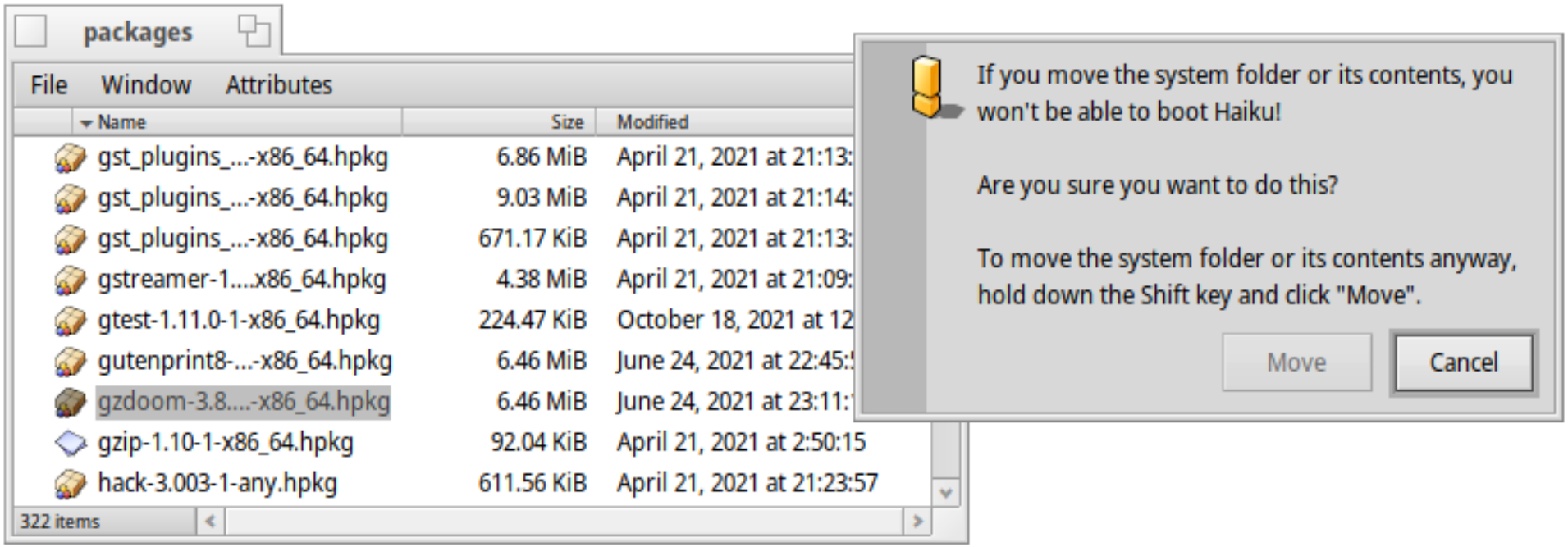
Demo!

There's another neat feature of Haiku's packaging system...

The Filesystem is Package-Aware



The Filesystem is Package-Aware



The Filesystem is Package-Aware

packages

File Window Attributes

Name	Size	Modified
gst_plugins_...-x86_64.hpkg	6.86 MiB	April 21, 2021 at 21:13:44
gst_plugins_...-x86_64.hpkg	9.03 MiB	April 21, 2021 at 21:14:32
gst_plugins_...-x86_64.hpkg	671.17 KiB	April 21, 2021 at 21:13:56
gstreamer-1....x86_64.hpkg	4.38 MiB	April 21, 2021 at 21:09:36
gtest-1.11.0-1-x86_64.hpkg	224.47 KiB	October 18, 2021 at 12:52:43
gutenprint8-...-x86_64.hpkg	6.46 MiB	June 24, 2021 at 22:45:55
gzip-1.10-1-x86_64.hpkg	92.04 KiB	April 21, 2021 at 2:50:15
hack-3.003-1-any.hpkg	611.56 KiB	April 21, 2021 at 21:23:57
haiku_datatr...-x86_64.hpkg	980.09 KiB	October 18, 2021 at 12:50:17

321 items

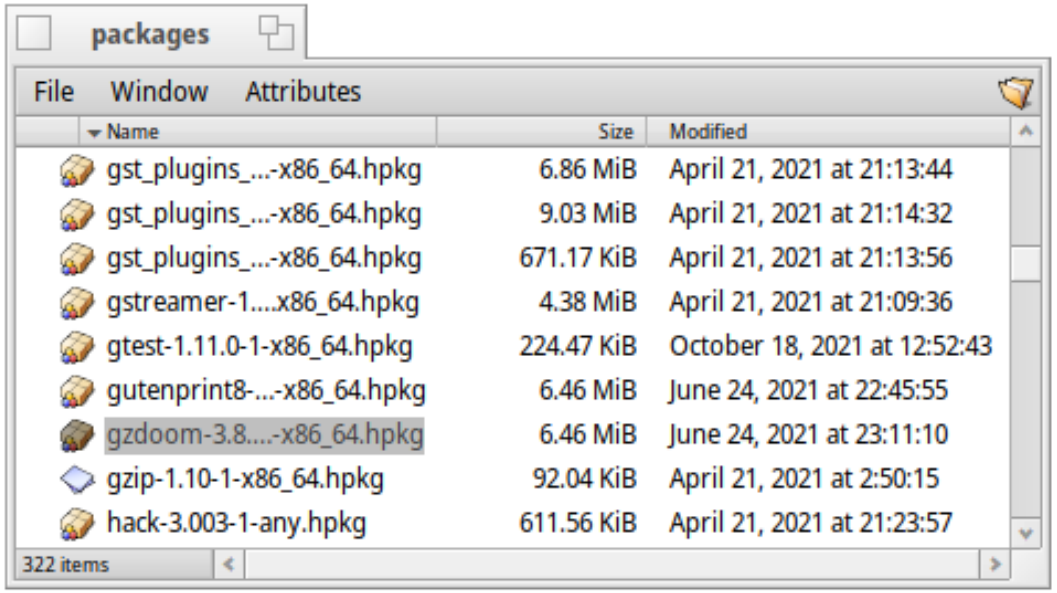
gzdoom-3.8....-x86_64.hpkg

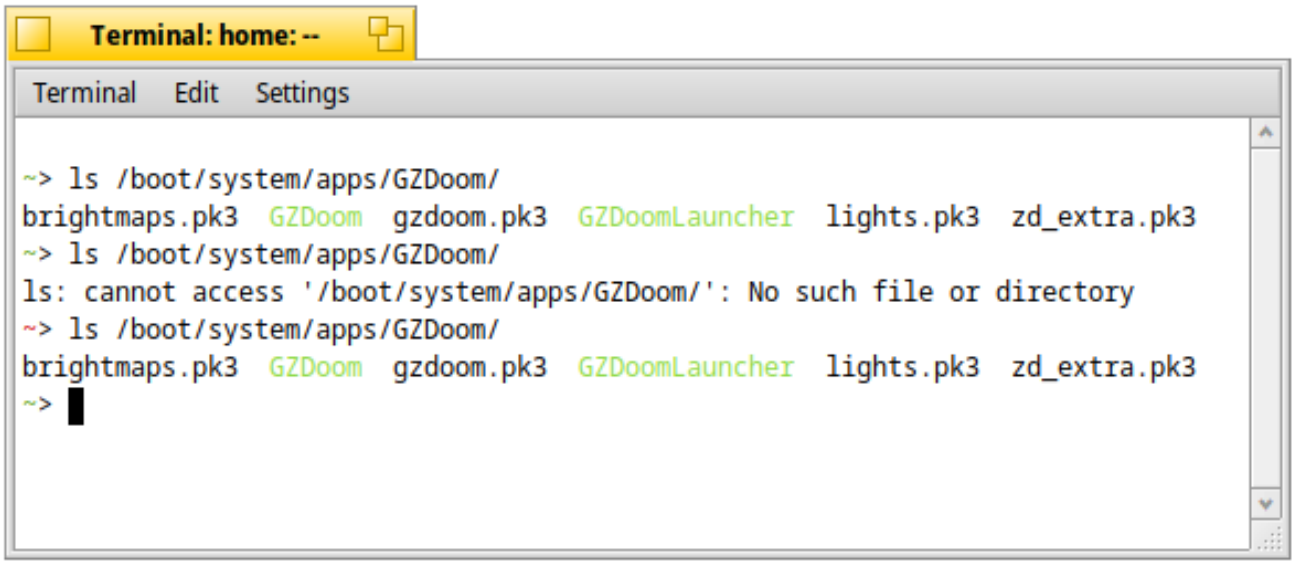
Terminal: home: --

Terminal Edit Settings

```
~> ls /boot/system/apps/GZDoom/
brightmaps.pk3  GZDoom  gzdoom.pk3  GZDoomLauncher  lights.pk3  zd_extra.pk3
~> ls /boot/system/apps/GZDoom/
ls: cannot access '/boot/system/apps/GZDoom/': No such file or directory
~>
```

The Filesystem is Package-Aware

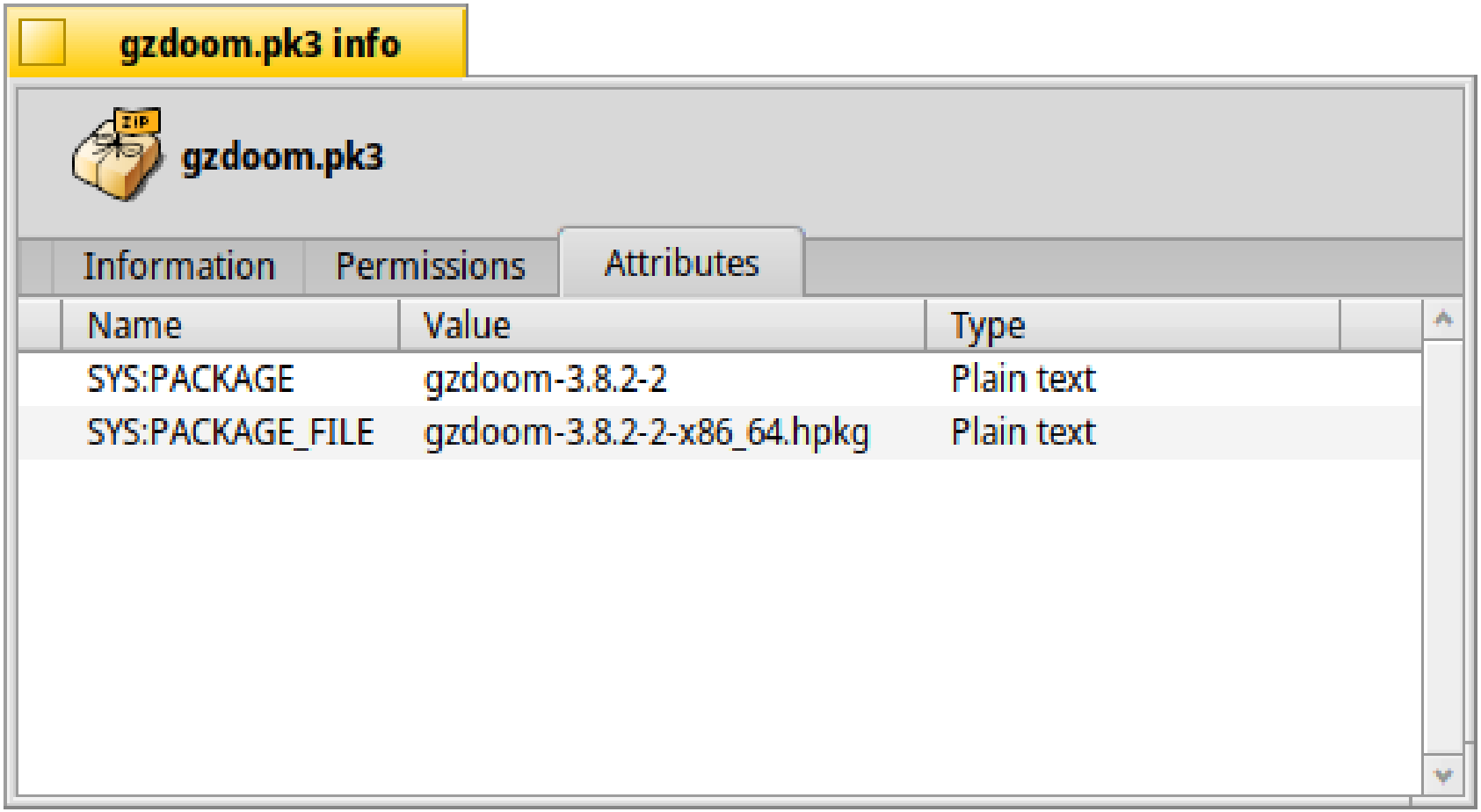




The Filesystem is Package-Aware

- Removing a package from the packages directory, `/boot/system/packages/`, uninstalls the package.
  - The system warns you that is is probably a bad idea.
  - It can be done anyway by holding the shift key.
- Putting the package back into the packages directory re-installs the package.

The Filesystem is Package-Aware



The BeFS attribute `SYS:PACKAGE_FILE` shows the package which is the source for a given file.

Questions?

Learn more about Haiku: `https://www.haiku-os.org/`

`https://github.com/haikuports/haikuports/`

IRC: #haiku on OFTC

`https://discuss.haiku-os.org`

`https://twitter.com/haikuOS`

`https://discord.gg/8KsjHbW`

Creating Packages

- Packages are created with a simple text file which contain metadata, typically in the root directory of the files to be packages, named "`.PackageInfo`"
- Required fields:
  - name
  - version
  - architecture – Specific architecture, list of architectures, or "any"
  - summary
  - description
  - packager – Person who packaged the software
  - licenses – Must be a license name known to the packager
  - provides – List of software names, executables, and/or libraries and version number
  - requires – List of required packages and/or libraries, optionally with version requirements

Creating Packages

- Directory contains the directories & files to be packages
  - `bin/` in the project will be mounted as `/boot/system/bin/package_name/`
  - `lib/` in the project will be mounted as `/boot/system/lib/package_name/`
- Command: `package create -C /path/to/dir/ -b Output.hpkg`
- Example: `https://github.com/rjzak/ghidra/blob/master_haiku64/.PackageInfo`

HaikuPorts

- Haiku has a vast collection of ported software, called HaikuPorts.
  - `https://github.com/haikuports/haikuports`
- HaikuPorts contains recipes to build software, and patches (if needed) to enable or improve Haiku support. These are built automatically and added to the repository.
- HaikuPorter helps build applications from HaikuPorts, and is used by the main package repository but can be run locally.
  - `https://github.com/haikuports/haikuporter`
  - `https://github.com/haikuports/haikuports.cross` for cross compiling for different architectures, with RISC-V getting a lot of attention recently