Building initrd images from rpms

Zbigniew Jędrzejewski-Szmek

zbyszek@in.waw.pl

DevConf2022, 28.01.2022

Why do we need an initrd?

a small file system that the boot loader passes to the kernel

Why do we need an initrd?

a small file system that the boot loader passes to the kernel

the purpose of the initrd is to mount the real root file system

Why do we need an initrd?

a small file system that the boot loader passes to the kernel

the purpose of the initrd is to mount the real root file system

initrd == initramfs

Status quo — dracut

Dracut — “generic initramfs intrastructure”

- configuration mechanism for deciding what is available in the initrd image (also a dependency mechanism with `check()`, `depends()`)
- create the image from files on the host (`instmods()`, `dracut_install()`, `inst()`, `inst_hook()`, `inst_rules()`)
- event-driven execution queue in initrd
- helpers to do various things in the initrd

“initrds over the years”

“initrds over the years”

stage I: “very special” — busybox with scripts, ...

“initrds over the years”

stage I: “very special” — busybox with scripts, ...
stage II: “special” — normal programs + custom event queue

“initrds over the years”

stage I: “very special” — busybox with scripts, ...
stage II: “special” — normal programs + custom event queue
stage III: “quasinormal” – normal programs + systemd + custom elements

“initrds over the years”

stage I: “very special” — busybox with scripts, ...
stage II: “special” — normal programs + custom event queue
stage III: “quasinormal” – normal programs + systemd + custom elements
stage IV: “normal“ — just systemd + normal services

Xorg –> Wayland, ca. 2008

technical debt
same people
long-term coexistence

Why is the initrd so special?

As far as the kernel is concerned, the initrd is just another file system.

The user-space is a bit different: `/init`, `/etc/initrd-release`

Why is the initrd so special?

As far as the kernel is concerned, the initrd is just another file system.

The user-space is a bit different: `/init`, `/etc/initrd-release`

Anything we would do in the initrd, we also need to do in the host:
storage
degraded storage
networking
network-based file systems and storage (nfs, iscsi, clevis)
fsck
emergency mode

Why is the initrd so special?

As far as the kernel is concerned, the initrd is just another file system.

The user-space is a bit different: `/init`, `/etc/initrd-release`

Anything we would do in the initrd, we also need to do in the host:
storage
degraded storage
networking
network-based file systems and storage (nfs, iscsi, clevis)
fsck
emergency mode

Nowadays all this functionality is implemented either using daemons and/or systemd units and/or various helpers.

What happens when a new file, a service, is added to a package?

packaging + dracut packaging

Overview of the new scheme

Dracut:

- configuration mechanism...
- create the image from files on the host
- event-driven execution queue...
- helpers ... in the initrd

Overview of the new scheme

Dracut:

- configuration mechanism...
- create the image from files on the host
- event-driven execution queue...
- helpers ... in the initrd

New scheme:

- a list of rpms

Overview of the new scheme

Dracut:

- configuration mechanism...
- create the image from files on the host
- event-driven execution queue...
- helpers ... in the initrd

New scheme:

- a list of rpms
- `dnf --installroot=... && cpio --create ...`

Overview of the new scheme

Dracut:

- configuration mechanism...
- create the image from files on the host
- event-driven execution queue...
- helpers ... in the initrd

New scheme:

- a list of rpms
- `dnf --installroot=… && cpio --create …`
- `systemd`

Overview of the new scheme

Dracut:

- configuration mechanism...
- create the image from files on the host
- event-driven execution queue...
- helpers ... in the initrd

New scheme:

- a list of rpms
- `dnf --installroot=... && cpio --create ...`
- `systemd`
- ordinary daemons

Advantaged of building the image directly from packages

- reliable installation: rpm is very good at doing what it does
- normal dependency mechanism

Advantaged of building the image directly from packages

- reliable installation: rpm is very good at doing what it does
- normal dependency mechanism
- we don't pull files from the host

Advantaged of building the image directly from packages

- reliable installation: rpm is very good at doing what it does
- normal dependency mechanism
- we don't pull files from the host
- images are immutable
- images are reproducible

Advantaged of building the image directly from packages

- reliable installation: rpm is very good at doing what it does
- normal dependency mechanism
- we don't pull files from the host
- images are immutable
- images are reproducible
- bash helpers $\rightarrow$ compiled programs
- developers don't need to learn another system

Advantaged of building the image directly from packages

- reliable installation: rpm is very good at doing what it does
- normal dependency mechanism
- we don't pull files from the host
- images are immutable
- images are reproducible
- bash helpers $\rightarrow$ compiled programs
- developers don't need to learn another system
- clear ownership of bugs
- any improvements are immediately shared

mkosi-initrd

- mkosi builds images from rpms
- mkosi-initrd uses mkosi to create a .cpio.zstd archive

---

mkosi-initrd

- mkosi builds images from rpms
- mkosi-initrd uses mkosi to create a .cpio.zstd archive

Some alternatives:

- osbuild
- kiwi-ng

Stages

I. local generation (just like dracut)

Stages

I. local generation (just like dracut)
II. generation in koji
(a curated set of initrd variants + extensions)

Stages

I. local generation (just like dracut)
II. generation in koji
(a curated set of initrd variants + extensions)
III. signing by Fedora keys (with opt-in signature verification)

Generation

```
sudo dnf install kernel-core

kernel-install add <version> <image>

mkosi -o initd.cpio.zstd
--build-env=KERNEL_VERSION=<version>
```

Size comparison

```
$ du -sh dracut-*.cpio.* mkosi-*.cpio.*
34M     dracut-5.13.4-200.fc34.x86_64.cpio.xz
62M     mkosi-5.13.4-200.fc34.x86_64.cpio.zstd
```

Size comparison

```
$ du -sh dracut-*.cpio.* mkosi-*.cpio.*
34M     dracut-5.13.4-200.fc34.x86_64.cpio.xz
62M     mkosi-5.13.4-200.fc34.x86_64.cpio.zstd
$ du -sh dracut-*.d/ mkosi-*.d/
77M     dracut-5.13.4-200.fc34.x86_64.d
165M    mkosi-5.13.4-200.fc34.x86_64.d
```

Size comparison

```
$ du -sh dracut-*.cpio.* mkosi-*.cpio.*
34M     dracut-5.13.4-200.fc34.x86_64.cpio.xz
62M     mkosi-5.13.4-200.fc34.x86_64.cpio.zstd
$ du -sh dracut-*.d/ mkosi-*.d/
77M     dracut-5.13.4-200.fc34.x86_64.d
165M    mkosi-5.13.4-200.fc34.x86_64.d
```

Some differences:
/lib/modules 5 MB vs. 37 MB
/usr/bin 8 MB vs. 18 MB
/usr/sbin 10 MB vs. 14 MB
/usr/lib64 41 MB vs. 51 MB
/usr/share 0.5 MB vs. 11 MB
(.../licenses 3 MB, .../zoneinfo 5 MB, .../pki 1 MB, .../terminfo 1 MB)
/etc 0.5 MB vs. 12 MB
(.../udev/hwdb.bin 9MB, .../pki 1 MB)

Extensions

systemd-sysext: extensions mounted with overlayfs

dm-verity + signatures

Building sysexts with mkosi

1. Mount an initramfs image somewhere
2. Mount an OverlayFS over it (upper layer empty)
3. `dnf install --installroot=… <packages for sysext>`
4. Create a file system image with upper layer only
5. (Optionally create partition dm-verity hash for it)
6. (Optionally sign the whole thing)

What works?

OK:

creation of initrd
integration with kernel scriptlets
building and use of sysexts
Fedora Server in QEmu with direct kernel boot
Normal laptops, LVM, LUKS
emergency mode without authentication
resume

Requires future work:

iscsi
switching back to initramfs for shutdown
firmware

Not tested:

fcoe, nfs, nbd, kdump, network syntax (ip=/ifname=/rd.route=/...) supported by dracut and systemd-network-generator, plymouth, network, raid, sshd, bluetooth, netconsole

Summary

Build initramfs images directly from system packages
Let systemd do the heavy lifting in the initrd
Do things in the initrd like on the host
Extend the initrd image using systemd-ext/OverlayFS
(Build initrd images and extensions in koji)
(Sign and verify all individual compoments)

Links

https://github.com/systemd/mkosi
https://github.com/systemd/mkosi-initrd
https://www.freedesktop.org/software/systemd/man/systemd-sysext.html
https://gitlab.com/cryptsetup/cryptsetup/-/wikis/DMVerity
https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/verity.html
https://www.kernel.org/doc/html/latest/filesystems/overlayfs.html
These slides:
https://github.com/keszybz/mkosi-initrd-talk