Getting your system to boot with initrd and initramfs

Ryan Curtin

LUG@GT
Goals

By the end of this presentation, hopefully, you should be able to:

- Know the basic history of initrd and initramfs, and why they exist
- Know how to configure GRUB / LILO to use an initrd or initramfs
- Unpack and look inside an initrd or initramfs
- Use standard tools to generate a new initrd or initramfs
- Modify an initrd and initramfs by hand or through configuration files
- Know how to configure a kernel to avoid using an initrd or initramfs
History of initrd and initramfs

- The idea of an initrd has been around almost as long as Linux itself, going back to the 1.xx kernels.
- It exists to load modules that are required by the kernel at boot-time, but not compiled into the kernel.
- The concept exists today in 2.6 kernels as initramfs and is used by distributions such as Fedora, Red Hat, Ubuntu, and Debian (there are more, of course).

http://ussg.iu.edu/hypermail/linux/kernel/9602/1289.html
Internals of an initrd

- The initrd is usually a cpio archive passed through gzip.
- Inside the archive is a straightforward simple directory hierarchy, similar to the standard Linux filesystem hierarchy.
- You can crack open an archive with

  ```bash
  $ gzip -dc initrd.img | cpio -idv
  ```

- `/linuxrc` and `/init` contain the script that is run to boot the system.
/linuxrc or /init - the initialization script

- /linuxrc or /init is the first executable started once the initrd or initramfs is loaded by the kernel
- /linuxrc is used in the older initrd
- /init is used in the newer initramfs
- This executable is usually a shell script; Debian uses /bin/sh but Red Hat uses /bin/nash; it does not matter which shell/interpreter is used
So what makes initrd and initramfs different?

- Both **initrd** and **initramfs** exist to solve the same problem; however, **initramfs** is used by modern 2.6 kernels whereas **initrd** was used by older 2.4 (and earlier) kernels.

- **initramfs** uses a dynamically-allocated RAM filesystem; **initrd** uses a statically-allocated RAM disk.

- A filesystem driver is required to read a **initrd** image at boot-time; **initramfs** requires only a lightweight ramfs driver, which is built-in by default in 2.6 kernels.

- **initramfs** makes NFS-mounted root filesystems easier; DHCP and logins may be necessary to mount an NFS share as root.
The Red Hat way

- The Red Hat utility for generating an initramfs is `mkinitrd`
- This utility seems to use the term `initrd` and `initramfs` interchangeably, as it produces an `initramfs`

General syntax:

```
mkinitrd -allow-missing -f initrd.img kernel-version
```

The `mkinitrd` command is referenced by

- `/sbin/new-kernel-pkg`
- `/sbin/installkernel`

`/etc/modprobe.conf` controls the modules that are put into the initramfs.
The Debian way

- The Debian utility for creating an initramfs is the slightly-more-aptly-named mkinitramfs
- Another utility is updateinitramfs, which can update the initramfs for all kernels on your machine
- yaird will also generate an initramfs, but it is less tested than the other Debian tools

A comparison of methods -

- yaird will generate a rather small initramfs by default; initramfs-tools will not
- initramfs-tools will create an initramfs no matter what; yaird will stop if it cannot be sure that the generated initramfs will work
- An initramfs generated by initramfs-tools will have an emergency shell if it fails; yaird does not include this unless you explicitly specify it
The Gentoo way

- A kernel configured by hand generally does not require an initramfs - the install guide encourages you to compile everything you need into the kernel
- A kernel created with genkernel will also create an initramfs
- mkinitrd is in the Portage tree; this could also be used
- Gentoo-Wiki has an extensive article about configuring your own initramfs -
  http://gentoo-wiki.com/HOWTO_Initramfs
By hand

When the initrd filesystem is done, it should contain (for proper startup)

- An `/init` script (necessary)
- The shell necessary to run the `/init` script and its dependencies
- Any modules that need to be loaded to mount the root filesystem

Pack up the filesystem with

```
$ find ./ > file_list
$ cpio -o < file_list > initrd.cpio
$ gzip initrd.cpio
$ rm file_list
```
Configuring your bootloader

- **GRUB - sample grub.conf excerpt**
  
  ```
  title Debian GNU/Linux, kernel 2.6.18-4-686
  root (hd0,0)
  kernel /vmlinuz-2.6.18-4-686 root=/dev/hda3 ro
  initrd /initrd.img-2.6.18-4-686
  ```

- **LILO - sample lilo.conf excerpt**
  
  ```
  image=/vmlinuz
  label=Linux
  read-only
  initrd=/initrd.img
  ```