Linux Early Userspace

initramfs, klibc, and...
putting things where they belong

H. Peter Anvin
Transmeta Corporation

Cast of Characters

- Al Viro (initramfs)
- H. Peter Anvin (klibc, protocol)
- Russell King (porting to klibc)
- Jeff Garzik (integrating with kernel build)

Linux 0.01 (1991)

- The root filesystem was mounted by the kernel, using a device number patched into the kernel binary via compile-time choice or hex editor (the `rdev` tool eventually replaces the hex editor for patching your kernel.)
- Root filesystem on disk. Period.
- *"Linux is user friendly. It's just selective as to who its friends are."*

Linux 0.9x (1992-1993)

- Kernel command line is added.
- `root=` command line option allows the root device to be set dynamically.
- Root device still on disk only.
- 0.98.3 (1992) – `root=` by number only.
  0.99.11 (1993) – `root=` by name.
- *The kernel now needs a device name to number mapping, which normally is provided by /dev.*

Linux 1.3.42 (1995)

- Allow the root filesystem to reside on NFS.
- *This means the kernel has to be able to configure TCP/IP networking, including the ability to talk RARP, BOOTP or DHCP.*

Linux 1.3.73 (1996)

- Initial ramdisk (`initrd`) mechanism introduced to let userspace deal with complex dependencies.
- Unfortunately, it still requires the initrd to specify a device number for the root device (what about network filesystems?)
- *The ramdisk code introduces a number of painful special cases in the buffer cache code.*
- *A filesystem image is hard to build on the fly.*

Linux 2.3.41 (2000)

- `pivot_root()` system call allows an initrd to play all kinds of games to gets its root filesystem mounted.
- ... *but pivot_root() has a bunch of odd special-case semantics, due to kernel threads starting up with the initrd as root.*
- *We'd like kernel threads to have* no *root, but that introduces special cases all over the place...*

We would like to...

- Eliminate special cases where possible.
- Replace kernel code with user space code...
  - Less likely to cause problems
  - Easier to write
  - Easier to customize
- Avoid problems like the initrd/kernel thread issue.

Linux 2.4.11 (2001)

- Introduce rootfs, a simple virtual filesystem using the ramfs code.
  - This makes ramfs mandatory, but it's very little code. In fact, making it mandatory lets filesystems like procfs use its code instead of adding its own.
- When the kernel starts, / is always rootfs. The "real" root is simply overmounted on top of the rootfs.
- Kernel threads start with / being rootfs.

Linux 2.4 (2001-2002)

- Change as much initialization code as we can to use standard system calls. Most standard system calls can be run from within the kernel once we have a root filesystem, and with rootfs, that can be very early.
- ... *but it's still running in the kernel, which means kernel programming rules apply, and that mistakes stay around forever.*

Linux 2.5 (2002-2003)

- Replace initrd with initramfs, which simply decodes a cpio archive of files onto the rootfs.
- This archive can be pregenerated, synthesized at boot time, or both. Multiple archives can be combined. We will probably allow it to be linked with the kernel.
- We should be able to remove initialization code from the kernel, and build a standard initramfs image.

klibc

- We need a lightweight C library that still provides a familiar development model.
  - glibc is overkill...
- klibc is < 20K as a shared i386 binary, and provides most basic C functionality and system calls. Some minimal porting is typically required.
- Shared, but not dynamic. Upgrading klibc requires a relink (and quite possibly a recompile.)

initramfs

- One or more cpio archives, possibly compressed, are archived onto the rootfs.
- Allows even a simple boot loader to construct images on the fly.
  - E.g. frequently requested: network boot loaders should save away all DHCP information...
- Open question: use a different ramfs? Makes garbage collection easier (unmount and it's gone.)

Writing early userspace code

- It's userspace. Normal C rules apply, however...
- Keep it small.
  - Make it possible to compile out features.
- Avoid external file dependencies.
- Line-oriented stdio input is very slow.
  - This can be fixed, but adds complexity and code size.
- `__KLIBC__` define makes it possible to write dual-mode code.

Candidates for moving to userspace

- Partition, RAID, and logical volume detection
- Detecting the root filesystem type
- nfsroot, including IP autoconfiguration
  - RARP, BOOTP, DHCP client
  - rpc to talk to the NFS mount daemon
- Replace kernel command-line handling?
  - The kernel command line is frequently too short for all the configuration information we'd like to pass

Current status (Jan 2003)

- initramfs
  - complete, integrated
  - not yet extensively tested, but seems to work
- klibc
  - basically complete
  - new features added on a demand basis
  - not yet ported to all architectures
  - not yet integrated

Current status (Jan 2003)

- User-space utilities
  - A number of utilities have been ported to or written for klibc, including the `ash` shell (55K static i386)
  - Ported tools are currently distributed with klibc
- Integration with kernel build
  - Necessary to allow tight coupling with kernel
  - In progress (Jeff Garzik)
  - Currently builds a basic initramfs, but not klibc