Booting Linux

The History and the Future

Werner Almesberger

EPFL ICA

Linux SYMPOSIUM
2000 Ottawa

Overview
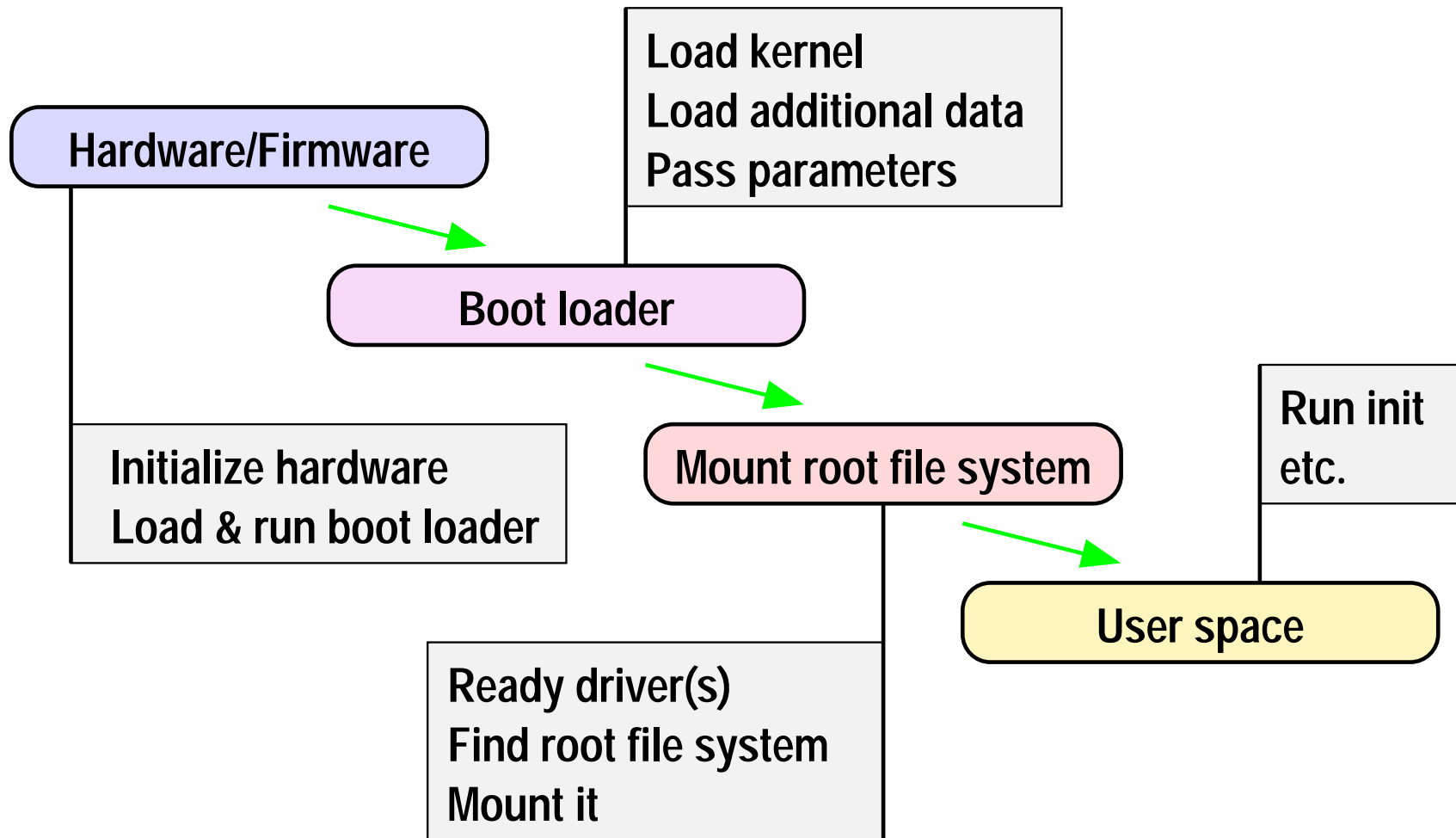
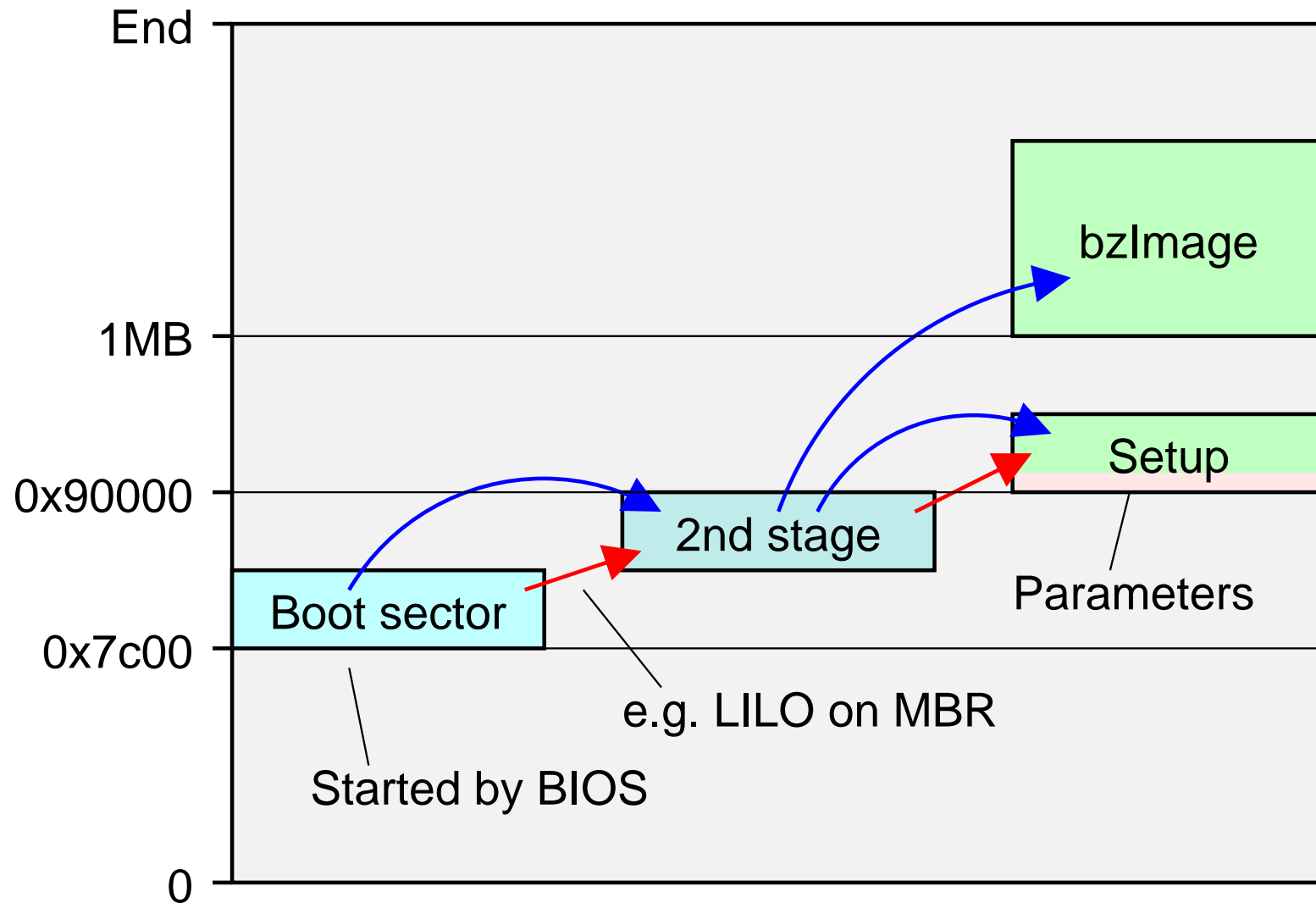- The i386 boot process
  - Overview
  - bzImage

- Evolution
  - Floppy boot sector, Shoelace
  - LILO, GRUB, etc.
  - Initial RAM disk (initrd)
  - Changing the root file system
  - Linux boots Linux
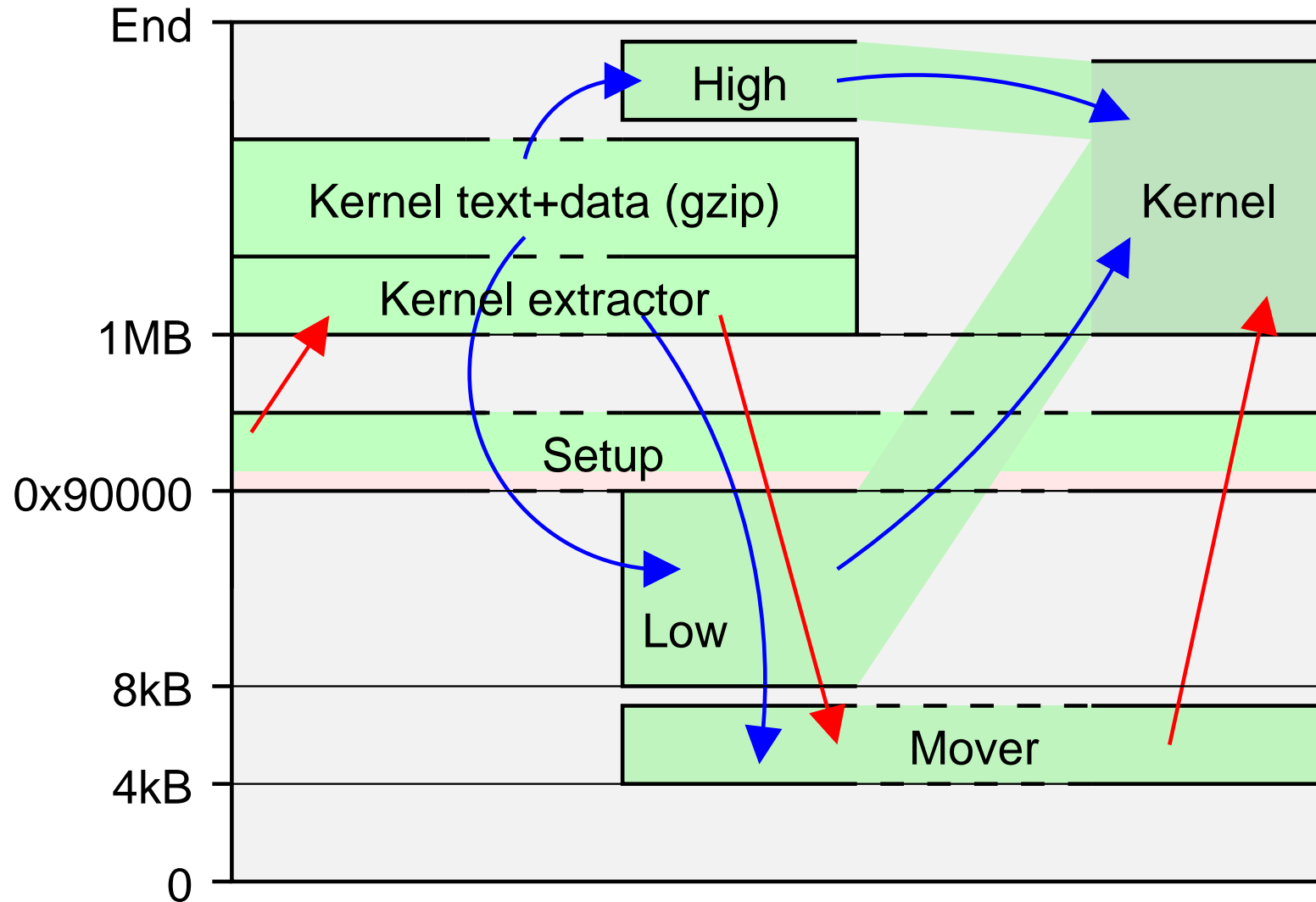
Boot process

Load kernel
Load additional data
Pass parameters

Hardware/Firmware

Boot loader

Run init
etc.

Initialize hardware
Load & run boot loader

Mount root file system

User space

Ready driver(s)
Find root file system
Mount it

Loading bzImage

End

bzImage

1MB

Setup

0x90000

2nd stage

Parameters

0x7c00

Boot sector

e.g. LILO on MBR

Started by BIOS
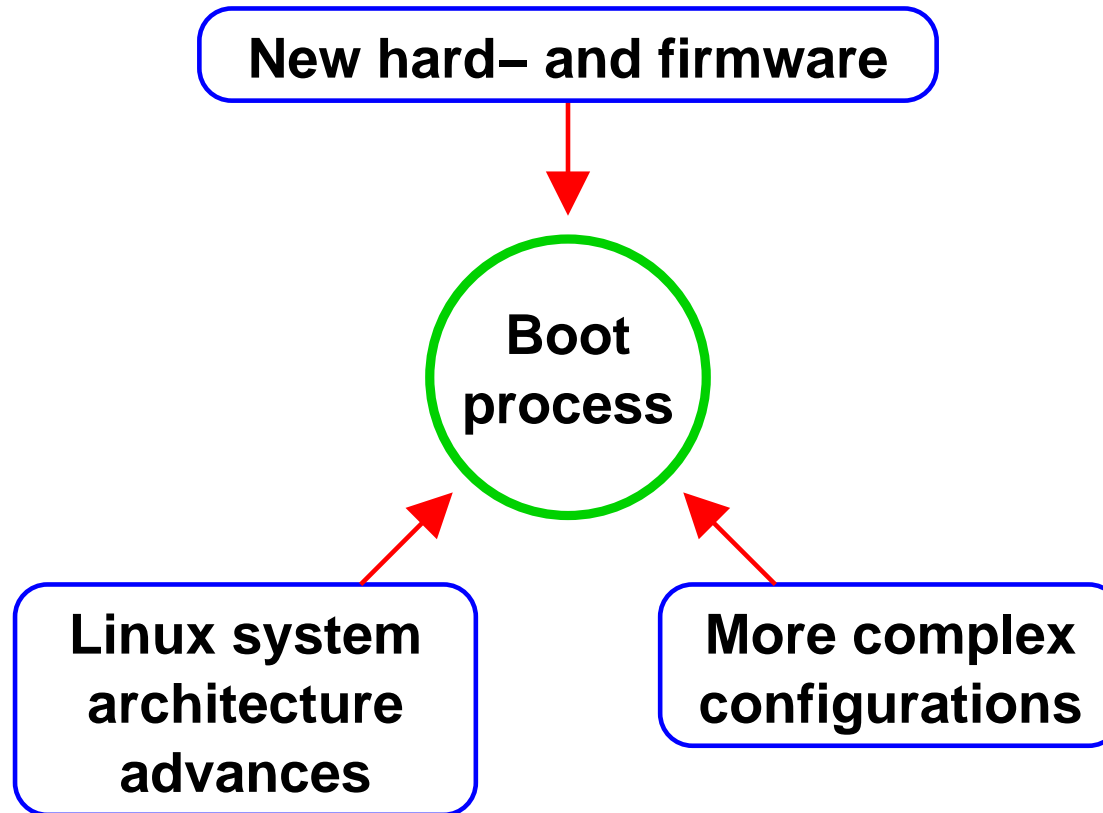
0

Starting bzImage

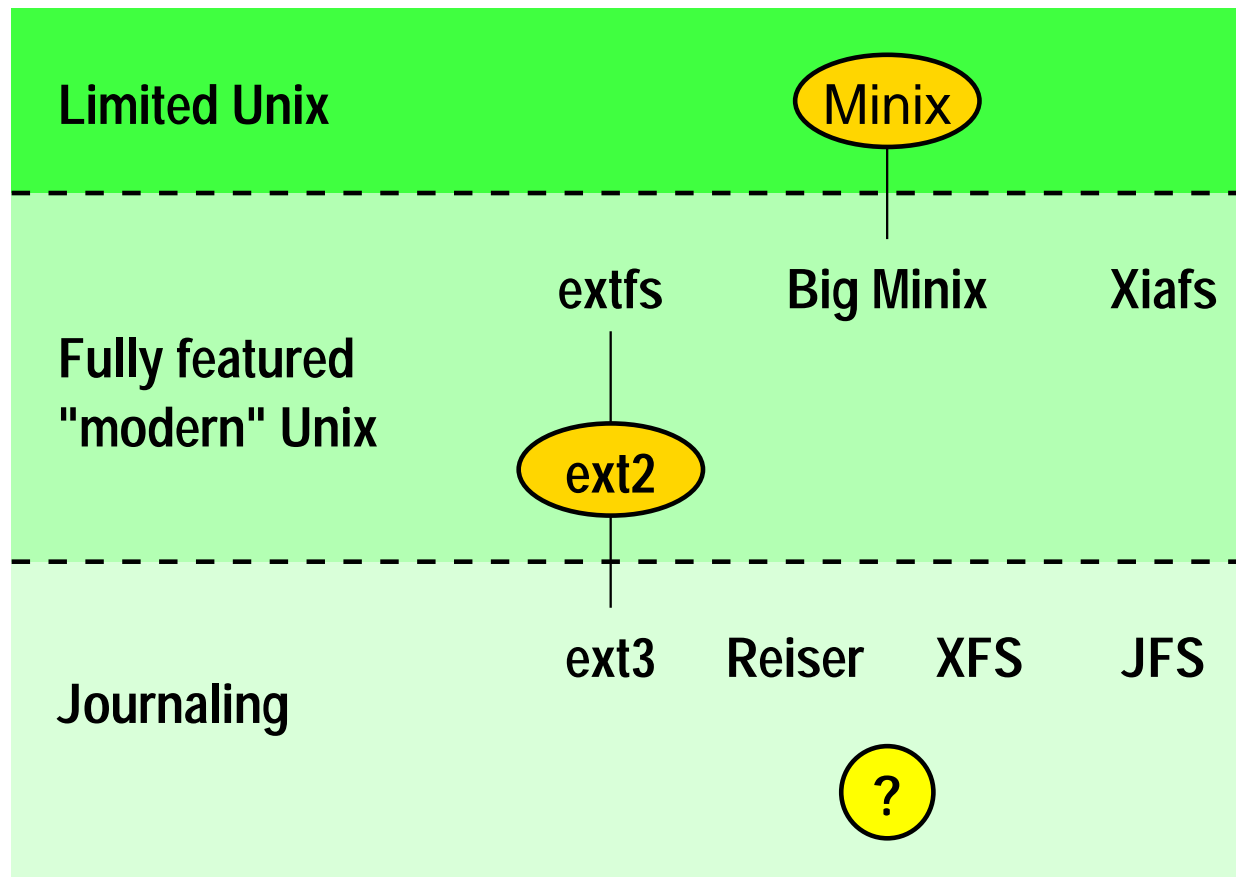# Challenges

Evolution: The humble beginnings

- Floppy boot sector
  - Boots from floppy disks
  - One kernel per floppy
  - Kernel parameter changes via `rdev`

- Shoelace
  - Boots from IDE disks
  - Requires Minix file system
  - Kernel parameter changes via `rdev`
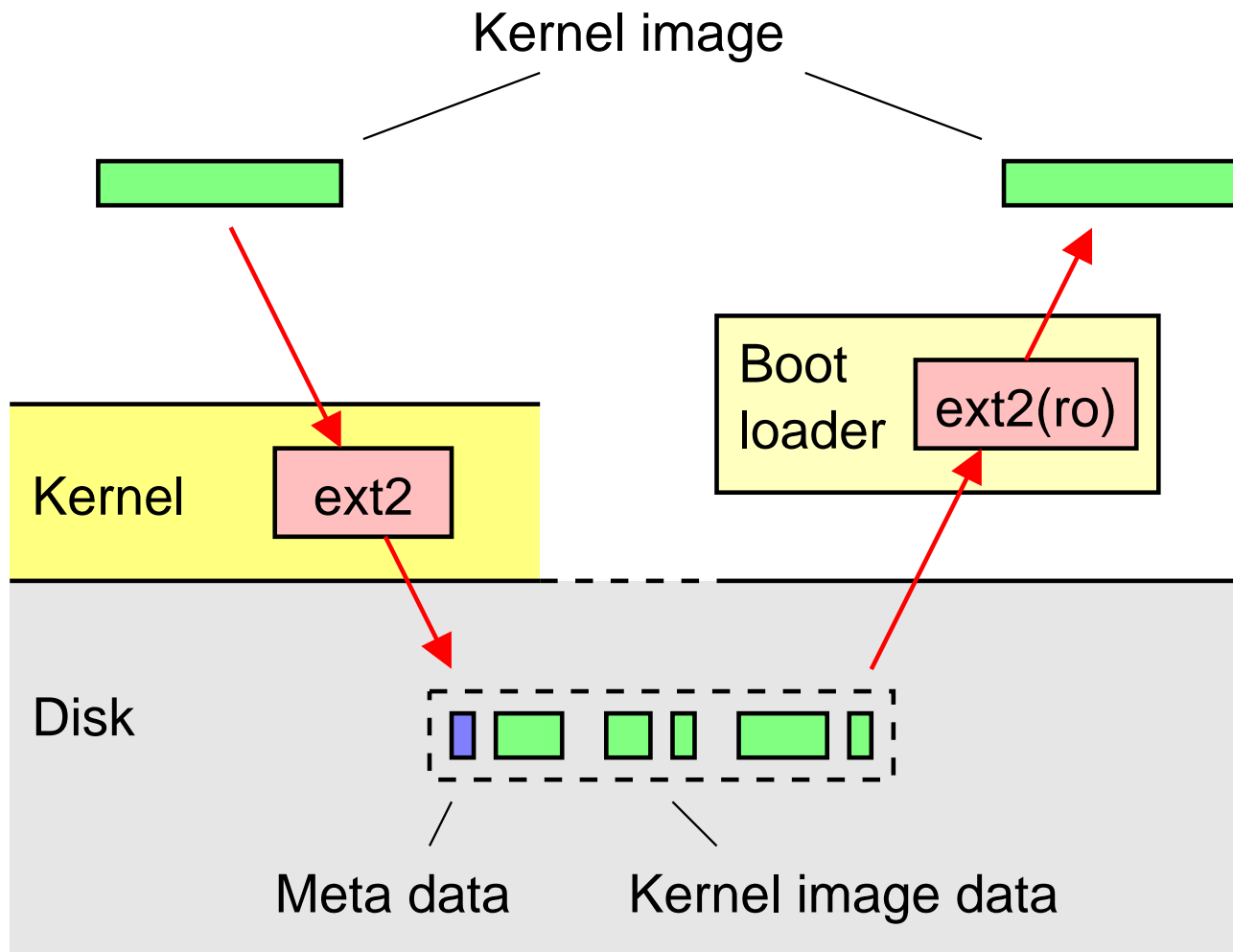 - Needs 3rd party boot selector

Problem: Non-Minix root

- Shoelace required Minix FS

  → needed extra partition with Minix FS

- Solution: teach boot loader to load from other FS

- Problem: there are so many of them …

- Solution: let the kernel do the mapping

LILO '92

- Files are "mapped"

- Boots from (almost) arbitrary file system

- Boots from floppy or any hard disk known to BIOS

- Boot command line

- Includes boot selector

Now maintained by John Coffman.

File system aware, e.g. GRUB

Kernel image

Kernel

ext2

Boot loader

ext2(ro)

Disk

Meta data   Kernel image data

File system unaware

Kernel image

Map

Boot loader

read list

Kernel

ext2

Disk

Meta data

Kernel image data

Meta data (map file)
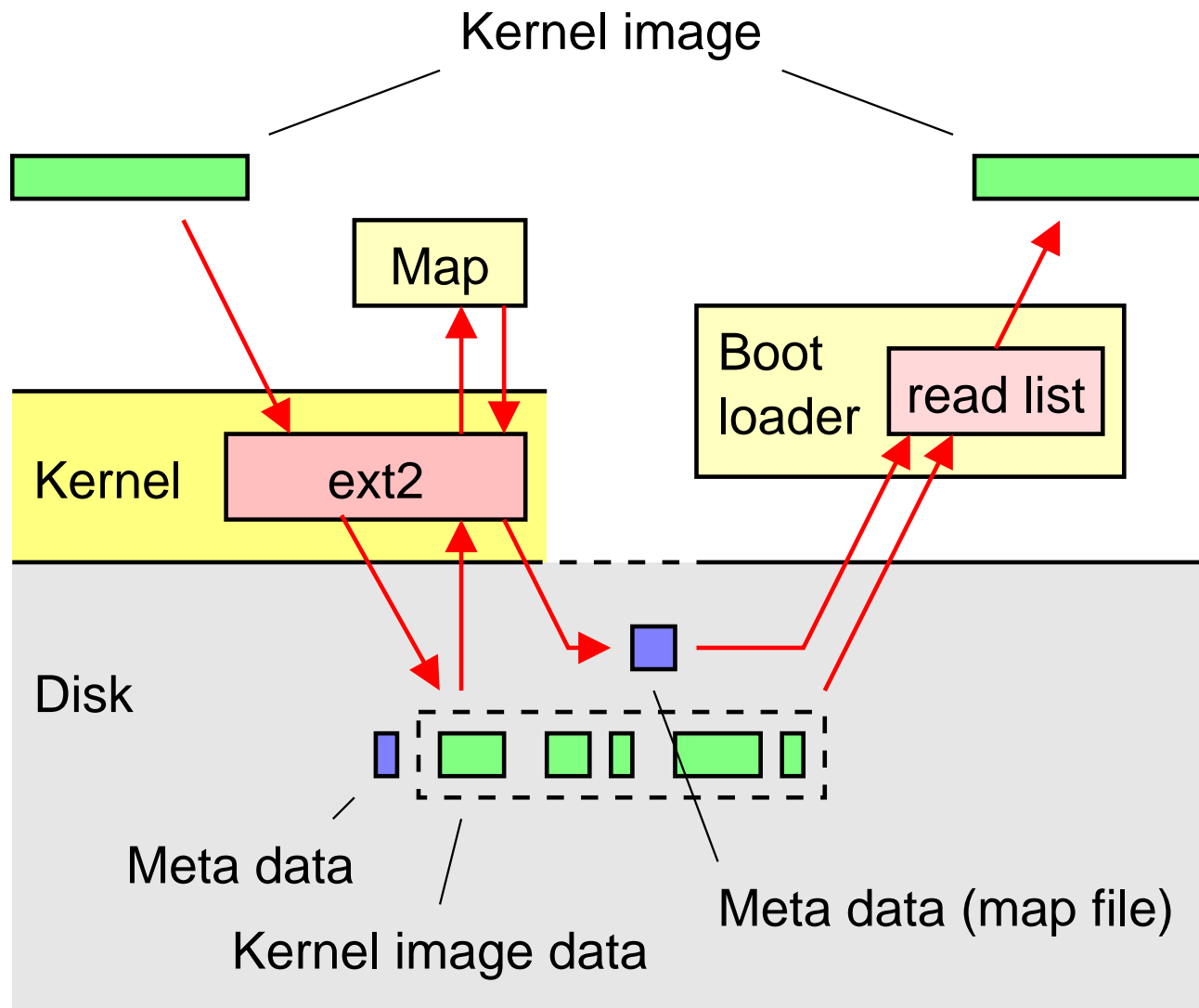
What's better ?

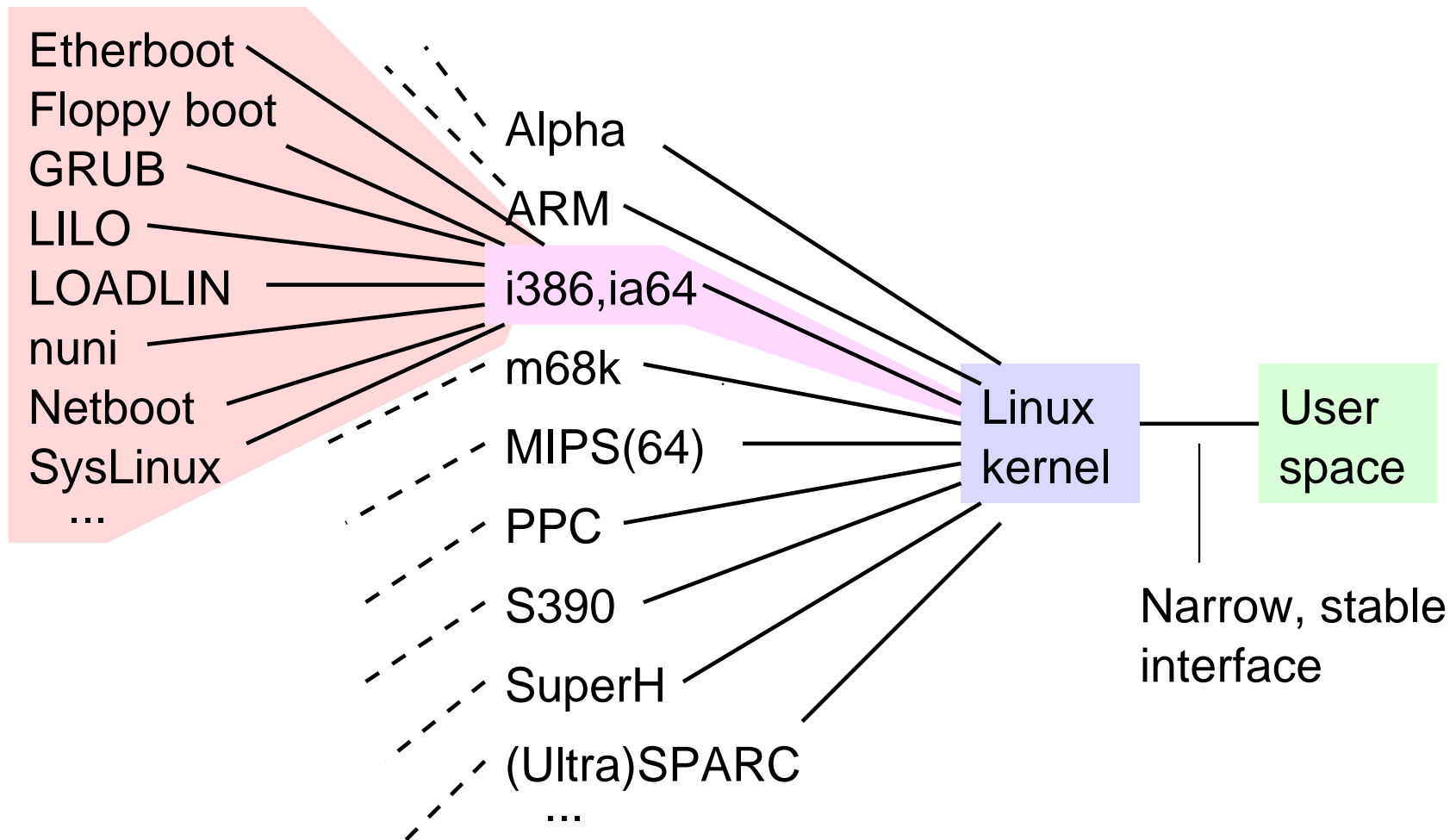- Most people don't notice
- Consistency check
- Hardware limitations

  (e.g. RAM disk, NFS)
- Configuration file

  (e.g. for command line)
- Autonomy
  - Pick any kernel
  - File system navigation
  - Editor, Web browser, … ?

Adding features

Etherboot
Floppy boot
GRUB
LILO
LOADLIN
nuni
Netboot
SysLinux
...

Alpha
ARM
i386,ia64
m68k
MIPS(64)
PPC
S390
SuperH
(Ultra)SPARC
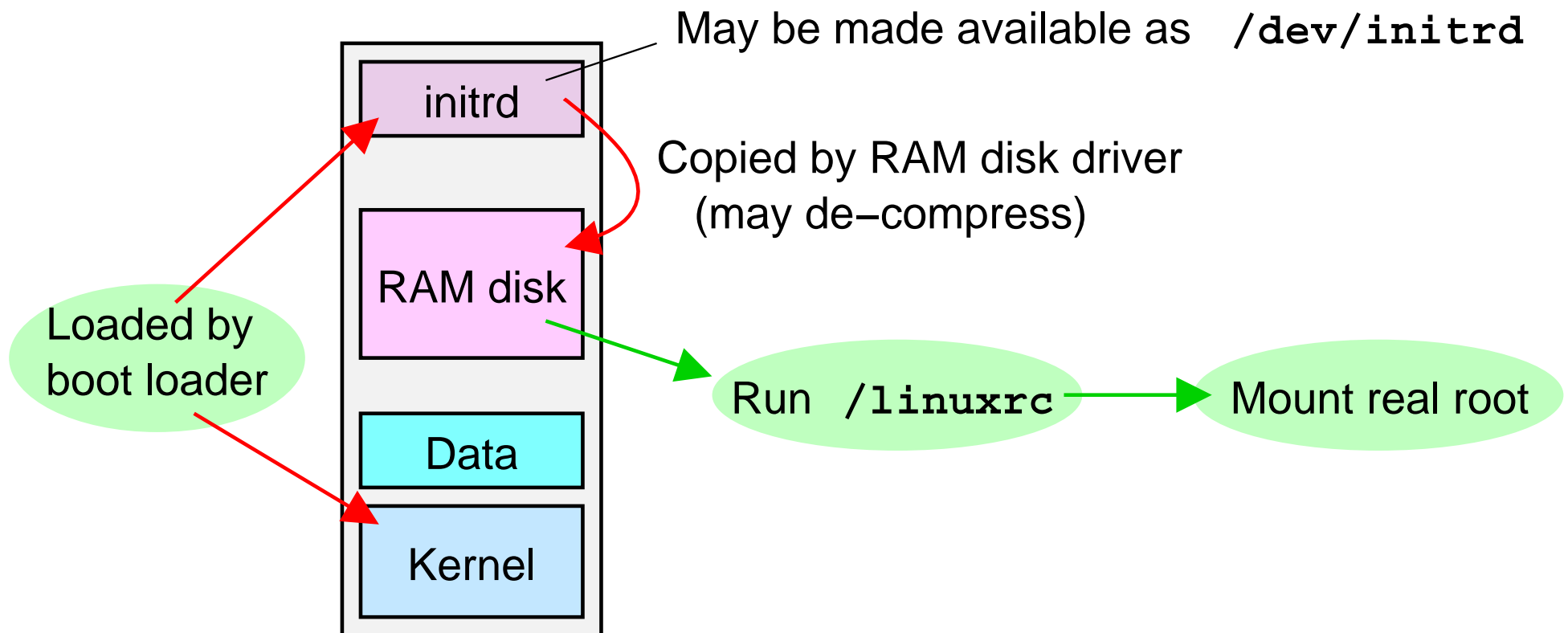...

Linux kernel

User space

Narrow, stable interface

Problem: Driver conflicts

- Hardware auto-detection is unreliable:
  - Hardware X may lock up when probing for hardware Y
  - Driver may mis-detect hardware X as Y, and crash
  - → "generic kernel" is not possible

- Solution: load driver modules when needed

- Problem: what if driver for root file system needs to be loaded ?

- Solution: boot loader loads RAM disk as temporary root

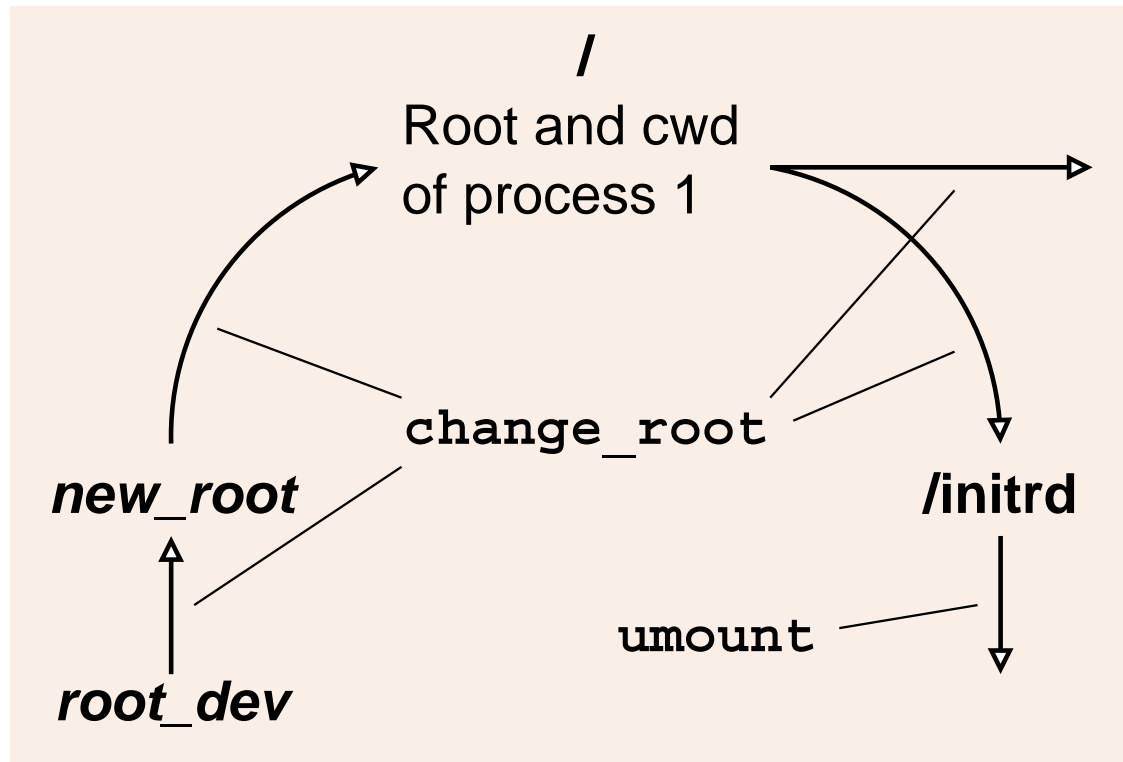# Initial RAM disk (1.3.73 3/96)

Designed in cooperation with Hans Lermen

May be made available as `/dev/initrd`

initrd

Copied by RAM disk driver
(may de-compress)

RAM disk

Loaded by
boot loader

Run `/linuxrc` → Mount real root

Data

Kernel

change_root (1.3.73, Mar '96)

/
Root and cwd
of process 1

change_root

new_root

/initrd

root_dev

umount

- Tightly integrated (uses lots of "magic")
- Only works for regular block devices (special case: NFS)
- Background processes may keep old root busy
- Can only be used once

pivot_root (2.3.41, Jan '00)

/
Root and cwd of
all processes

pivot_root

new_root (usually cwd of the
current process)

old_root

mount   **umount**
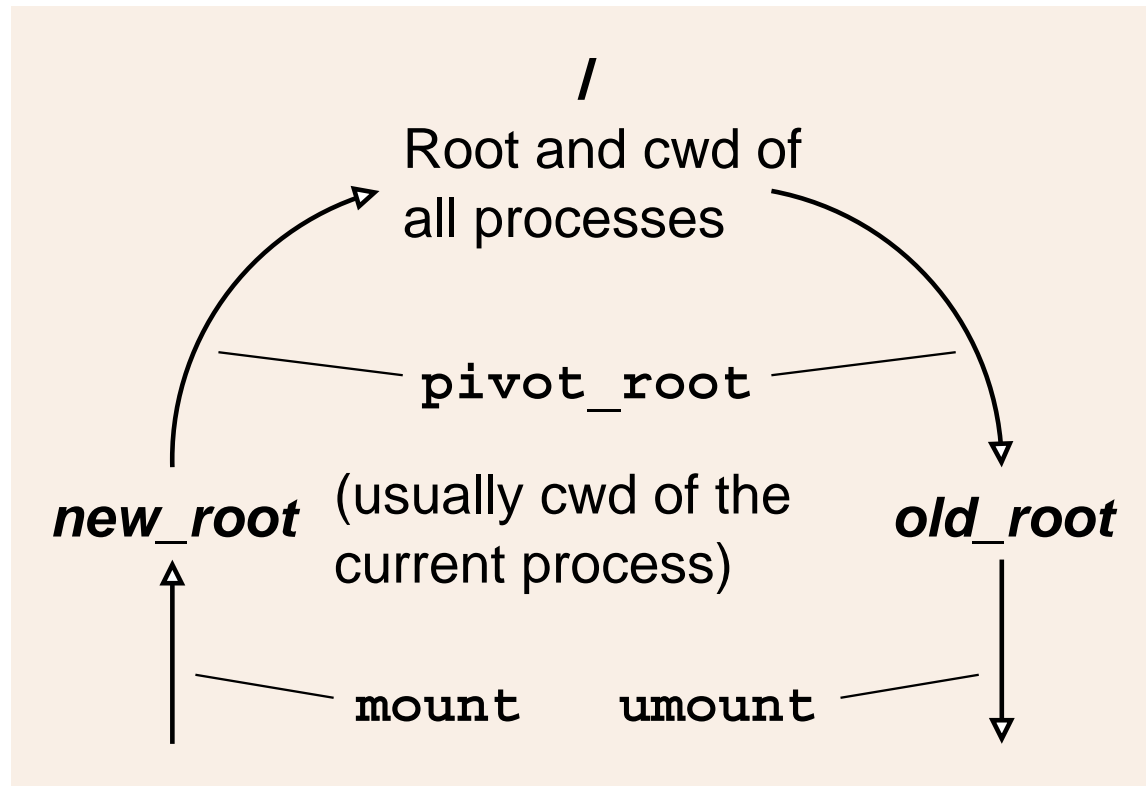
🐧 Works with any file system

🐧 Can be repeated

🐧 Very ugly solution for demons

Union mounts (future)

Root and cwd of
all processes

mount

/

umount

remount

new_root   **old_root**
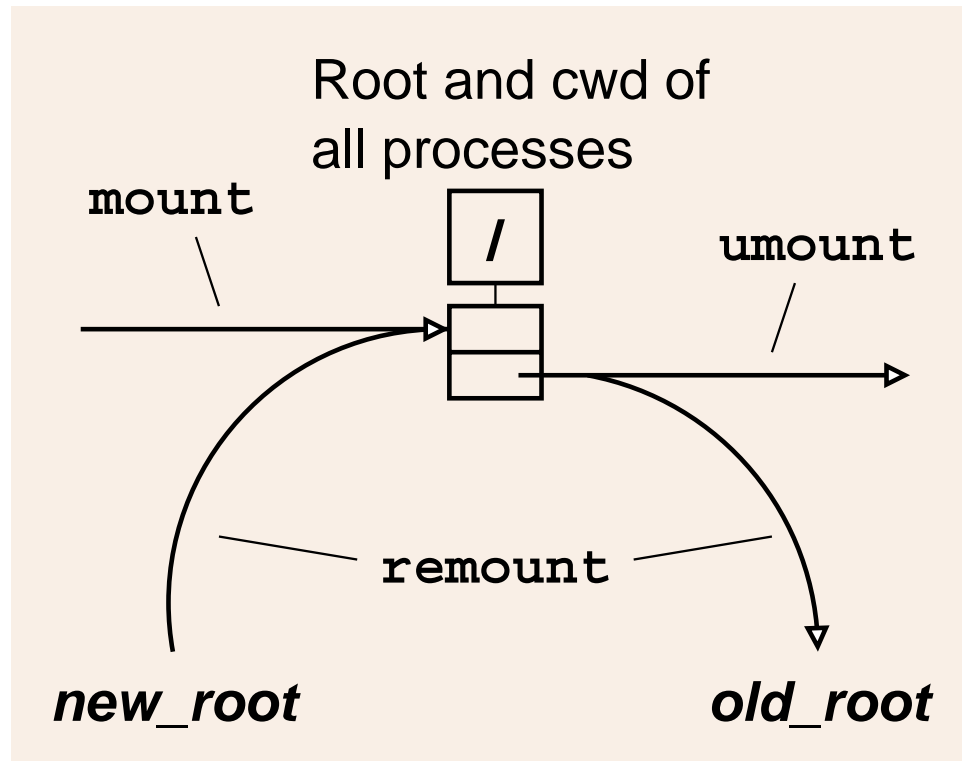
🐧 Designed by VFS guru Alexander Viro
🐧 As generic as `pivot_root`
🐧 Clean solution for demons
🐧 Will probably work with existing system calls and tools

Problem: Exotic kernel source

New complex file systems:

- No blocks: Tails (e.g. in ReiserFS), Database-oriented
- Blocks move: Journaling
- No direct mapping: RAID

Kernels stored at unusual places:

- TFTP, NFS, HTTP, HTTPS, ...
- Character device
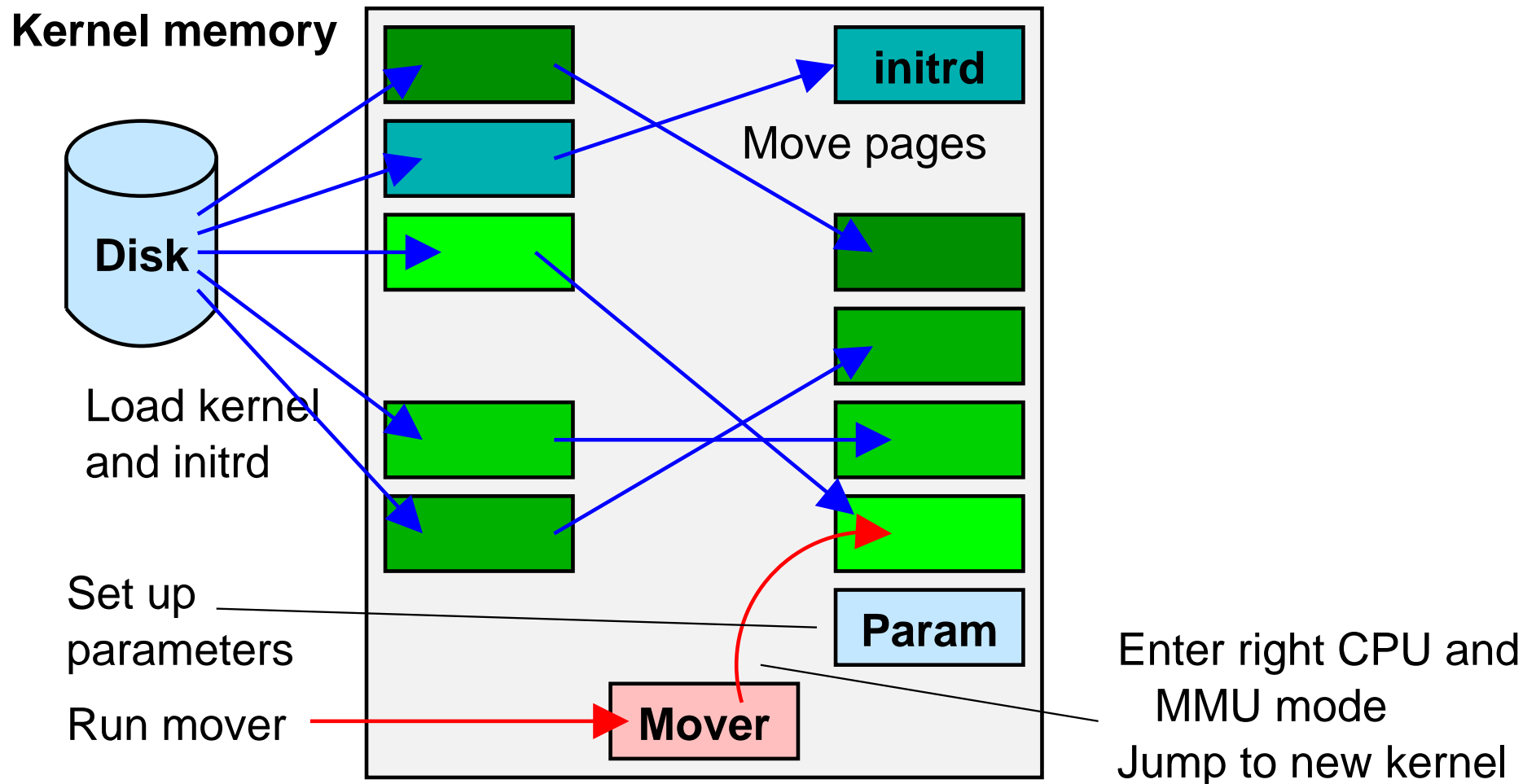
→ just loading the kernel gets harder

Solution: let the Linux system do the loading

Linux boots Linux (1)

- Simple boot loader loads boot kernel

- Driver setup using initrd

- Maybe user interaction (e.g. boot shell)

- Loads real kernel

Linux boots Linux (2)

Kernel memory

initrd

Move pages

Disk

Load kernel
and initrd

Set up
parameters

Param

Run mover   **Mover**

Enter right CPU and
MMU mode
Jump to new kernel

Linux boots Linux (3)

The time is ripe:

- LOBOS (Ron Minnich)
- bootimg (Werner Almesberger)
- Two Kernel Monte (Erik Hendriks)

More fun with booting:
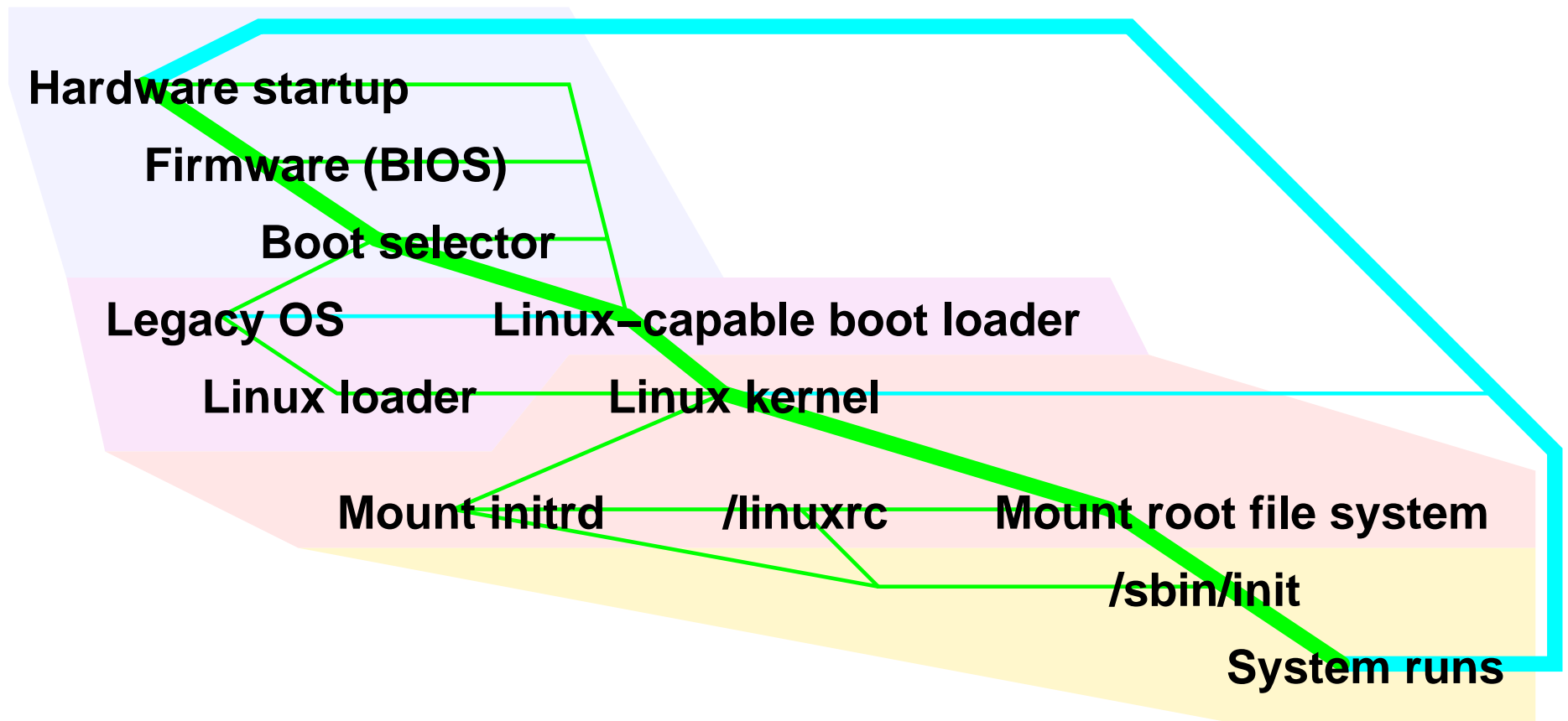
- High-speed reboot (ca. 15 sec)
- LinuxBIOS
- Crash Dump

  Problem:

  ○ BIOS clears memory on reboot

  ○ Cannot use current drivers (or BIOS)

  Solution:

  ○ Keep small system inside current system

  ○ Reset drivers and write dump on crash

The whole truth

Hardware startup

Firmware (BIOS)

Boot selector

Legacy OS   Linux–capable boot loader

Linux loader   Linux kernel

Mount initrd   /linuxrc   Mount root file system

/sbin/init

System runs

Future

- Union root mount

- Early freeing of initrd pages
  - Reduce peak memory usage

- Linux boots Linux
  - Small C library or boot shell
  - Parameter passing

- Better support for RAID, etc.
  - Specialized loaders
  - Partial convergence for simple loaders
  - Extended data location information from kernel

Links

- Directory:

  `ftp://icaftp.epfl.ch/pub/people/almesber/booting/`

- This presentation:

  `ols-booting-slides.ps.gz`

  `ols-booting-slides.tar.gz`

- Paper "Booting Linux: The History and the Future":

  `bootinglinux-current.ps.gz`

  `bootinglinux-current.tar.gz`