Linux Kernel Tinification

Josh Triplett <josh@joshtriplett.org>
Jamey Sharp <jamey@minilop.net>

April 16, 2012
Linux kernel size

As of Linux 3.4-rc2

- defconfig: 4.9MB
  - Boots and runs on most current machines.
As of Linux 3.4-rc2

- defconfig: 4.9MB
  - Boots and runs on most current machines.
- allnoconfig: 803kB
  - No filesystems, disks, or networking
Linux kernel size

As of Linux 3.4-rc2

- defconfig: 4.9MB
  - Boots and runs on most current machines.
- allnoconfig: 803kB
  - No filesystems, disks, or networking
- Manually shrunk further: 339kB
  - No keyboard, VGA, or printk
Still too big

- Embedded systems with 256kB–1MB flash parts
Still too big

- Embedded systems with 256kB–1MB flash parts
- coreboot/LinuxBIOS

Let's use Linux as the bootloader instead
Still too big

- Embedded systems with 256kB–1MB flash parts
- coreboot/LinuxBIOS
- x86 boot track (65024 bytes)
  - GRUB and other bootloaders fit just fine
  - GRUB: tinier core, piles of copied/rewritten drivers
  - Let’s use Linux as the bootloader instead
Your project

- Make Linux smaller
Your project

- Make Linux smaller
- Make a useful Linux kernel fit in as little space as possible
Your project

- Make Linux smaller
- Make a useful Linux kernel fit in as little space as possible
- In a clean way suitable for inclusion in upstream Linux
Potential approaches

- Remove unused code
Potential approaches

- Remove unused code
- Make more features optional
Potential approaches

- Remove unused code
- Make more features optional
  - Preemptive scheduling
Potential approaches

- Remove unused code
- Make more features optional
  - Preemptive scheduling
  - Users and permissions
Potential approaches

- Remove unused code
- Make more features optional
  - Preemptive scheduling
  - Users and permissions
  - Filesystem write support

- Users and permissions
- Filesystem write support
- Userspace and syscalls
- uCLinux
- Write tiny alternative drivers
- BIOS-based disk support

- Link-time optimization
  - Your idea here
Potential approaches

- Remove unused code
- Make more features optional
  - Preemptive scheduling
  - Users and permissions
  - Filesystem write support
  - Userspace and syscalls
Potential approaches

• Remove unused code
• Make more features optional
  • Preemptive scheduling
  • Users and permissions
  • Filesystem write support
  • Userspace and syscalls
• uclinux
Potential approaches

- Remove unused code
- Make more features optional
  - Preemptive scheduling
  - Users and permissions
  - Filesystem write support
  - Userspace and syscalls
- uclinux
- Write tiny alternative drivers
  - BIOS-based disk support
Potential approaches

- Remove unused code
- Make more features optional
  - Preemptive scheduling
  - Users and permissions
  - Filesystem write support
  - Userspace and syscalls
- uclinux
- Write tiny alternative drivers
  - BIOS-based disk support
- Link-time optimization
Potential approaches

- Remove unused code
- Make more features optional
  - Preemptive scheduling
  - Users and permissions
  - Filesystem write support
  - Userspace and syscalls
- uclinux
- Write tiny alternative drivers
  - BIOS-based disk support
- Link-time optimization
- Your idea here
Success criteria

- Required: Implement at least one of the “ambitious” size-reduction goals
Success criteria

- Required: Implement at least one of the “ambitious” size-reduction goals
- Implement whatever else you can
Success criteria

- Required: Implement at least one of the “ambitious” size-reduction goals
- Implement whatever else you can
- Stretch goal: Hit the 64kB target to put Linux in the x86 boot track
Success criteria

- Required: Implement at least one of the “ambitious” size-reduction goals
- Implement whatever else you can
- Stretch goal: Hit the 64kB target to put Linux in the x86 boot track
- Anything else you want to do to make Linux more suitable for embedded systems.
Resources

- Us
  - Experienced kernel developers
  - We’ll help you get your code upstream
Resources

• Us
  • Experienced kernel developers
  • We’ll help you get your code upstream
• Andi Kleen, Intel
  • LTO expertise
Resources

- Us
  - Experienced kernel developers
  - We’ll help you get your code upstream
- Andi Kleen, Intel
  - LTO expertise
- Kernel developers in Portland and elsewhere
Resources

- Us
  - Experienced kernel developers
  - We’ll help you get your code upstream
- Andi Kleen, Intel
  - LTO expertise
- Kernel developers in Portland and elsewhere
- LKML
  - Send patches for review as you complete them
  - Merge windows align perfectly
Previous Capstones

- Virtual Ring Routing for Linux
- X.org nested driver
  - Currently upstream, X.org is deprecating a pile of old code in favor of the results of this project.
- New X.Org server for MacOS X
  - Apple’s X.org developer planning to ship it ASAP, hoping to replace the current server in the future.
Summary

- Easy progress indicator: how much smaller have you made the kernel?
Summary

- Easy progress indicator: how much smaller have you made the kernel?
- Hack on an Open Source project

Questions?
Josh Triplett <josh@joshtriplett.org>
Jamey Sharp <jamey@minilop.net>
Summary

- Easy progress indicator: how much smaller have you made the kernel?
- Hack on an Open Source project
- Do something highly visible

Questions?
Josh Triplett <josh@joshtriplett.org>
Jamey Sharp <jamey@minilop.net>
Summary

- Easy progress indicator: how much smaller have you made the kernel?
- Hack on an Open Source project
- Do something highly visible
- Kernel hackers *always* have jobs
Summary

• Easy progress indicator: how much smaller have you made the kernel?
• Hack on an Open Source project
• Do something highly visible
• Kernel hackers *always* have jobs

Questions?

Josh Triplett <josh@joshtriplett.org>
Jamey Sharp <jamey@minilop.net>