05-1 The Kernel

It all started with...

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix –

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat(same physical layout of the file-system (due to practical reasons)among other things).

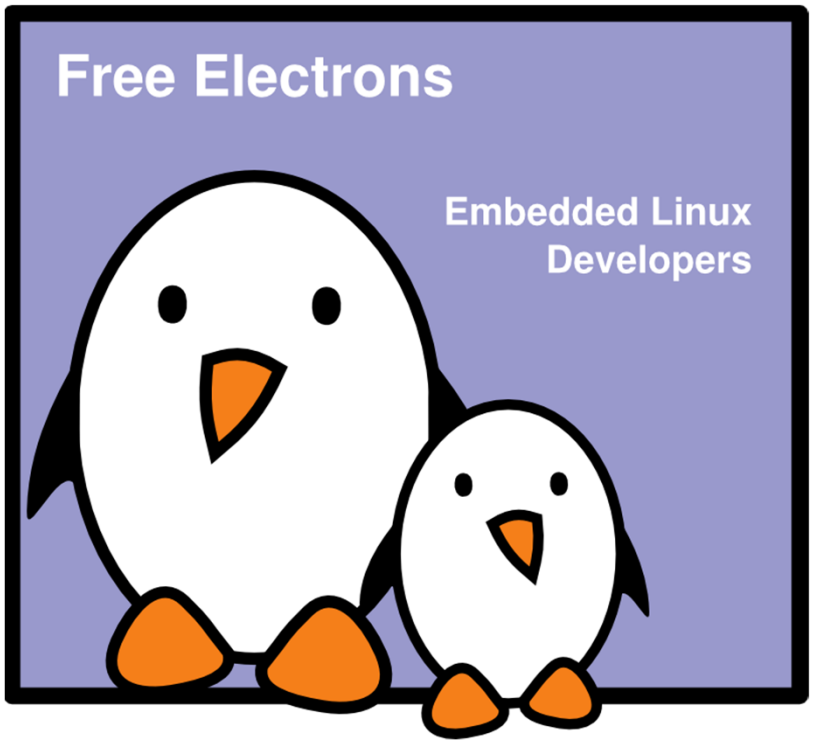
I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

Free Electrons

Linux kernel introduction

Michael Opdenacker
Thomas Petazzoni
Free Electrons



Latest update: 9/28/2017,
Document sources, updates and translations:
http://free-electrons.com/docs/kernel-intro
Corrections, suggestions, contributions and translations are welcome!

Embedded Linux driver development

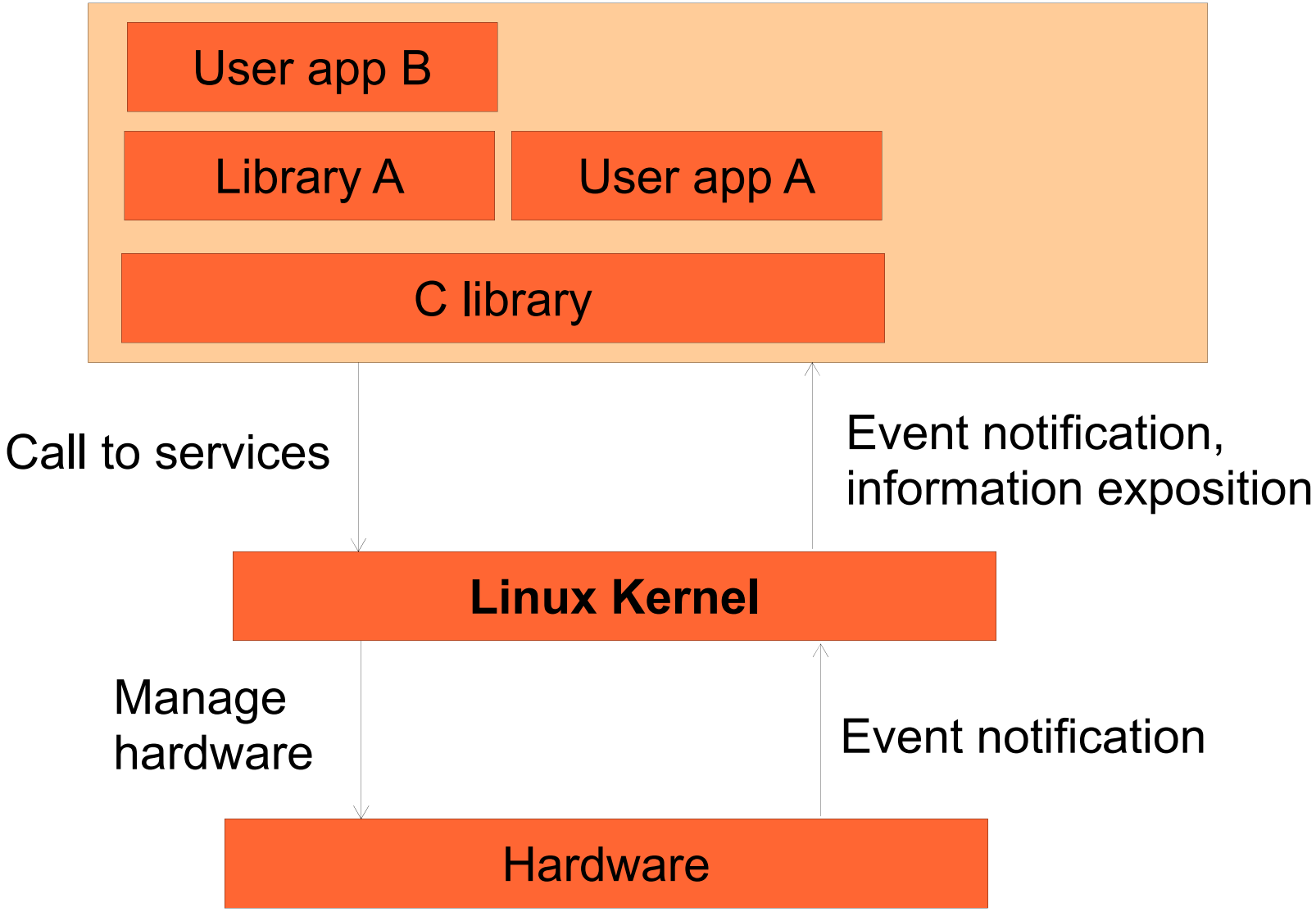
Kernel overview

Linux features

History

- The Linux kernel is one component of a system, which also requires libraries and applications to provide features to end users
- The Linux kernel was created as a hobby in 1991 by a Finnish student, Linus Torvalds
- Linux quickly started to be used as the kernel for free software operating systems
- Linus Torvalds has been able to create a large and dynamic developer and user community around Linux
- Nowadays, hundreds of people contribute to each kernel release, individuals or companies big and small

Linux kernel in the system



Supported hardware architectures

2.6.31 status

What's the current version?

4.13.4

- See the …/arch/ directory in the kernel sources
- Minimum: 32 bit processors, with or without MMU, and gcc support
- 32 bit architectures (…/arch/ subdirectories)
  arm, avr32, blackfin, cris, frv, h8300, m32r, m68k, m68knommu, microblaze, mips, mn10300, parisc, s390, sparc, um, xtensa
- 64 bit architectures:
  alpha, ia64, sparc64

How did I find it?

kernel.org

- 32/64 bit architectures
  powerpc, x86, sh
- Find details in kernel sources: …/arch/<arch>/Kconfig or …/Documentation/<arch>/

The Linux Kernel Archives

About Contact us FAQ Releases Signatures Site news

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel: 4.13.4

mainline:	4.14-rc2	2017-09-24	[tarball]		[patch]	[inc. patch]	[view diff]	[browse]
stable:	4.13.4	2017-09-27	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
stable:	4.12.14 [EOL]	2017-09-20	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	4.9.52	2017-09-27	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	4.4.89	2017-09-27	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	4.1.44	2017-09-14	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	3.18.72 [EOL]	2017-09-27	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	3.16.48	2017-09-15	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	3.10.107	2017-06-27	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	3.4.113	2016-10-26	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	3.2.93	2017-09-15	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
linux-next:	next-20170928	2017-09-28						[browse]

Other resources

Cgit
Patchwork
Linux.com
Wikis
Kernel Mailing Lists
Linux Foundation
Bugzilla
Mirrors
Kernel Planet

Social

Site Atom feed
Releases Atom Feed
Linux on Google+

What version is the Bone running?

```
bone$ uname -a
Linux bone-0834 4.4.84-ti-r120 #1 SMP
Sun Aug 27 03:11:07 UTC 2017 armv7l
GNU/Linux
```

System calls

What are examples?

- The main interface between the kernel and userspace is the set of system calls
- About ~300 system calls that provides the main kernel services
- This interface is
  calls can be ad
- This system cal
  library, and use
  make a system
  corresponding

File and device operations, networking operations, inter-process communication, process management, memory mapping, timers, threads, synchronization primitives, etc.

Pseudo filesystems

- Linux makes system and kernel information available in user space through **pseudo filesystems**, (also called **virtual filesystems**
- Pseudo filesystems allow applications to see directories and files that do not exist on any real storage: they are created and updated on the fly by the kernel
- The two most important pseudo file systems are
  - **`proc`**, usually mounted on **`/proc`**: Operating system related information (processes, memory management parameters...)
  - **`sysfs`**, usually mounted on **`/sys`**: Representation of the system as a set of devices and buses. Information about these devices.

/proc details

A few examples:

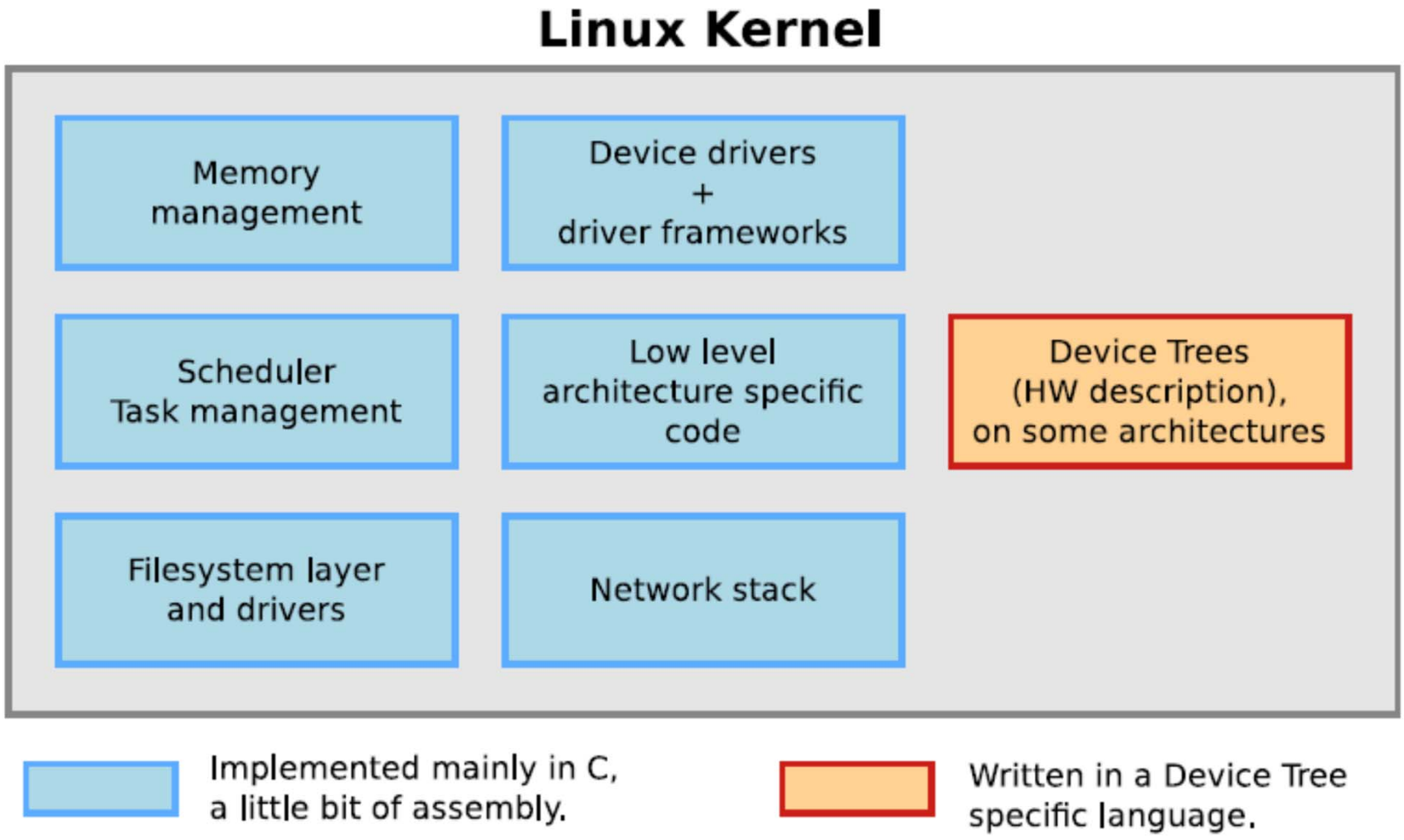
- /proc/cpuinfo: processor information
- /proc/meminfo: memory status
- /proc/version: kernel version and build information
- /proc/cmdline: kernel command line
- /proc/<pid>/environ: calling environment
- /proc/<pid>/cmdline: process command line

Lots of details about the /proc interface are available in …/Documentation/filesystems/proc.txt
(some 1700 lines) in the kernel sources.

... and many more! See by yourself!

```
beagle$ ls –F /proc
1/     18/    259/   508/  757/  asound/       ioports       schedstat
10/    1857/  26/    509/  76/   buddyinfo     irq/          scsi/
1064/  1863/  27/    54/   769/  bus/          kallsyms      self@
11/    1880/  2753/  549/  77/   cgroups       keys          slabinfo
1106/  1881/  28/    55/   771/  cmdline       key-users     softirqs
12/    1882/  29/    553/  774/  config.gz     kmsg          stat
1200/  19/    3/     573/  78/   consoles      kpagecgroup   swaps
1232/  1983/  30/    58/   785/  cpu/          kpagecount    sys/
1235/  2/     31/    59/   786/  cpuinfo       kpageflags    sysrq-trigger
1236/  20/    32/    6/    79/   crypto        loadavg       sysvipc/
1242/  2004/  4339/  60/   8/    devices       locks         thread-self@
1246/  2006/  4353/  61/   80/   device-tree@  meminfo       timer_list
1247/  21/    4704/  62/   809/  diskstats     misc          timer_stats
1248/  22/    4712/  620/  81/   driver/       modules       tty/
14/    2257/  4714/  63/   812/  execdomains   mounts@       uptime
15/    2258/  4789/  7/    82/   fb            mtd           version
1593/  2274/  4795/  73/   878/  filesystems   net@          vmallocinfo
16/    23/    4813/  74/   9/    fs/           pagetypeinfo  vmstat
1689/  24/    4815/  748/  939/  interrupts    partitions    zoneinfo
17/    25/    5/     75/   apm   iomem         sched_debug
```

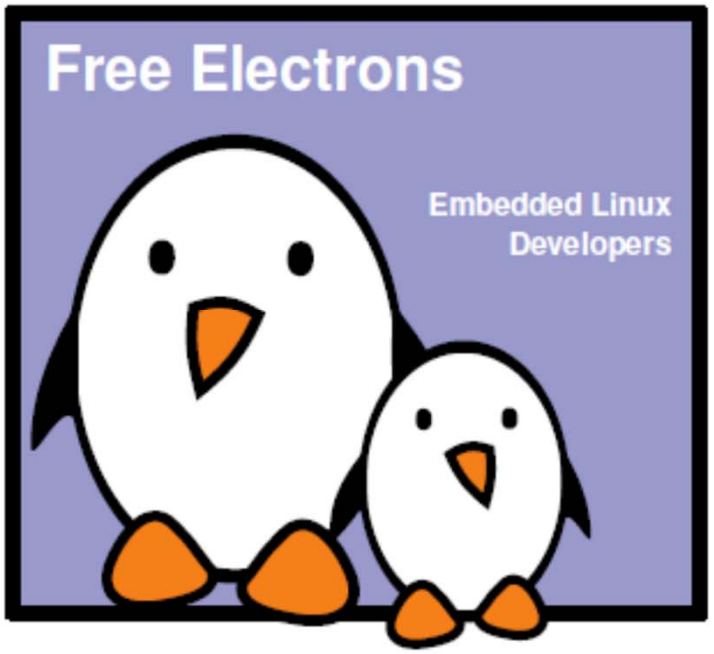
Inside the Linux kernel



Embedded Linux usage

Embedded Linux Kernel Usage

Free Electrons



What's new in each Linux release?

```
commit 3c92c2ba33cd7d666c5f83cc32aa590e794e91b0
Author: Andi Kleen <ak@suse.de>
Date:   Tue Oct 11 01:28:33 2005 +0200

    [PATCH] i386: Don't discard upper 32bits of HWCR on K8

    Need to use long long, not long when RMWing a MSR. I think
    it's harmless right now, but still should be better fixed
    if AMD adds any bits in the upper 32bit of HWCR.

    Bug was introduced with the TLB flush filter fix for i386

    Signed-off-by: Andi Kleen <ak@suse.de>
    Signed-off-by: Linus Torvalds <torvalds@osdl.org>
...
```

- The official list of changes for each Linux release is just a huge list of individual patches!
- Very difficult to find out the key changes and to get the global picture out of individual changes.
- Fortunately, a summary of key changes with enough details is available on http://wiki.kernelnewbies.org/LinuxChanges

Kernel Hacking
Kernel Newbies Frontpage
Kernel Hacking
Kernel Documentation
Kernel Glossary
FAQ
Found a Bug?
Kernel Changelog
Upstream Merge Guide

Projects
Kernel Janitors
Kernel Mentors
Kernel Projects

Community
Why a community?
Web Forum
Regional Kernelnewbies
Personal Pages
Upcoming Events

References
Mailing Lists
Related Sites
Programming Links

Wiki
Recent Changes
Site Editors
Navigation Menu
Site Statistics
Tips for Editors
Find Page
Help Contents
Hosted by WikiWall

This site Web
Google Search

Immutable Page Info Attachments More Actions:

LinuxChanges

Changes done in each Linux kernel release. Other places to get news about the Linux kernel are LWN kernel status, H-Online, or the Linux Kernel mailing list (there is a web interface in www.lkml.org). List of changes of older releases can be found at LinuxVersions. If you're going to add something here look first at LinuxChangesRules!

You can discuss the latest Linux kernel changes on the New Linux Kernel Features Forum.

Linux 3.16 released has been released on Sun, 3 Aug

Summary: This release improves performance with the support dynamically switch the clock frequency on Nvidia cards, it adds support for mapping user space memory into the GPU on Intel devices, XFS has a free inode btree for faster inode allocation, ARM 64 kernels can be used as EFI stubs, TCP Fast Open is supported in IPv6, some radeon devices have better performance thanks to improved power management support , Intel Cherryview graphics are supported, and control groups have gained an optional Unified Hierarchy mode, new drivers and many other small improvements have also been added.

1. Prominent features
    1. Nvidia graphics performance improvements, initial support for GK20A devices and GK110B
    2. Intel graphic driver allows mapping of user pages into video memory
    3. Unified Control Group hierarchy
    4. XFS free inode btree, for faster inode allocation
    5. Allow booting ARM 64 kernels as EFI stubs
    6. TCP Fast Open server mode on IPv6 support
    7. Intel Cherryview graphics support
    8. Radeon performance improvements through improved APU power management have been enabled in some APUs
2. Drivers and architectures
3. Core
4. Memory management
5. Block layer
6. Power management
7. File systems
8. Networking
9. Virtualization
10. Tracing/perf
11. Security
12. Crypto
13. Other news sites that track the changes of this release

1. Prominent features

1.1. Nvidia graphics performance improvements, initial support for GK20A devices and GK110B

Nouveau, the opensource driver for Nvidia graphic GPUs, has gained support for allowing to change the frequency of the GPU from the BIOS predefined values. This feature (which for now needs to be enabled manually) improves performance noticeably. The Nvidia GPUs that got reclocking support in this release are those with nv40, nvaa, and nve0 clock types.

This release also adds initial (but incomplete) support for NVidia GK20A graphic chips, found in Tegra K1 SoC; and GK110B devices

Code: commit, commit, commit

1.2. Intel graphic driver allows mapping of user pages into video memory

LinuxChanges

Kernel Hacking
- Kernel Newbies Frontpage
- Kernel Hacking
- Kernel Documentation
- Kernel Glossary
- FAQ
- Found a Bug?
- Kernel Changelog
- Upstream Merge Guide

Projects
- Kernel Janitors
- Kernel Mentors
- Kernel Projects

Community
- Why a community?
- Web Forum
- Regional Kernelnewbies
- Personal Pages
- Upcoming Events

References
- Mailing Lists
- Related Sites
- Programming Links

Wiki
- Recent Changes
- Site Editors
- Navigation Menu
- Site Statistics
- Tips for Editors
- Find Page
- Help Contents
- Hosted by WikiWall

LinuxChanges

Changes done in each Linux kernel release. Other places to get news about the Linux kernel are LWN kernel status, H-Online, or the Linux Kernel mailing list (there is a web interface in www.lkml.org). List of changes of older releases can be found at LinuxVersions. If you're going to add something here look first at LinuxChangesRules!

You can discuss the latest Linux kernel changes on the New Linux Kernel Features Forum.

Linux 4.7 was released on Sun, 24 Jul 2016.

Summary: This release adds support for the recent Radeon RX 480 GPUs, support for parallel pathname lookups in the same directory, a new experimental 'schedutils' frequency governor that should be faster and more accurate than existing governors, support for the EFI 'Capsule' mechanism for upgrading firmware, support for virtual USB Devices in USB/IP to make emulated phones behave as real USB devices, a new security module '`LoadPin`' that ensures that all kernel modules are loaded from the same filesystem, an interface to create histograms of events in the ftrace interface, support for attaching BPF programs to kernel tracepoints, support for callchains of events in the perf trace utility, stable support for the Android's sync_file fencing mechanism, and many other improvements and new drivers.

1. Prominent features
    1. Support for Radeon RX480 GPUs
    2. Parallel directory lookups
    3. New 'schedutil" frequency governor
    4. Histograms of events in ftrace
    5. perf trace calls stack
    6. Allow BPF programs to attach to tracepoints
    7. EFI 'Capsule' firmware updates
    8. Support for creating virtual USB Device Controllers in USB/IP
    9. Android's sync_file fencing mechanism considered stable
    10. LoadPin, a security module to restrict the origin of kernel modules
2. Core (various)
3. File systems
4. Memory management
5. Block layer
6. Security
7. Tracing, perf, BPF
8. Virtualization
9. Networking
10. Architectures
11. Drivers
    1. Graphics
    2. Storage
    3. Staging
    4. Networking
    5. Audio
    6. Input devices: Tablets, touch screens, keyboards, mouses
    7. TV tuners, webcams, video capturers
    8. USB
    9. Serial Peripheral Interface (SPI)
    10. Watchdog
    11. Serial
    12. ACPI, EFI, cpufreq, thermal, Power Management

LinuxChanges

Login

Kernel Hacking
- Kernel Newbies Frontpage
- Kernel Hacking
- Kernel Documentation
- Kernel Glossary
- FAQ
- Found a Bug?
- Kernel Changelog
- Upstream Merge Guide

Projects
- Kernel Janitors
- Kernel Mentors
- Kernel Projects

Community
- Why a community?
- Web Forum
- Regional Kernelnewbies
- Personal Pages
- Upcoming Events

References
- Mailing Lists
- Related Sites
- Programming Links

Wiki
- Recent Changes
- Site Editors
- Navigation Menu
- Site Statistics
- Tips for Editors
- Find Page
- Help Contents
- Hosted by WikiWall

LinuxChanges

Changes done in each Linux kernel release. Other places to get news about the Linux kernel are LWN kernel status, H-Online, or the Linux Kernel mailing list (there is a web interface in www.lkml.org). List of changes of older releases can be found at LinuxVersions. If you're going to add something here look first at LinuxChangesRules!

You can discuss the latest Linux kernel changes on the New Linux Kernel Features Forum.

Linux 4.12 has been released on Sun, 2 July 2017.

Summary: This release includes a new BFQ I/O scheduler which provides a much better interactive experience; it also includes preliminary support for Radeon RX Vega graphic cards and support for USB Type-C connectors; improvements to the live kernel patching feature, support for Intel IMSM's Partial Parity Log which allows to close the RAID5 write hole; support for exposing OpenChannel SSDs as device blocks, and another I/O scheduler, Kybe that allows to configure a latency target for reads and writes, ,

1. Prominent features
    1. Preliminary Radeon Vega support
    2. USB Type-C support
    3. New BFQ I/O scheduler for a more reponsive desktop
    4. New Kyber I/O scheduler
    5. Progress in Live kernel patching
    6. Add support for Intel IMSM's Partial Parity Log
    7. Expose OpenChannel SSDs as device blocks
2. Core (various)
3. File systems
4. Memory management
5. Block layer
6. Tracing and perf tool
7. Virtualization
8. Cryptography
9. Security
10. Networking
11. Architectures
12. Drivers
    1. Graphics
    2. Storage
    3. Drivers in the Staging area

Location of kernel sources

- The official versions of the Linux kernel, as released by Linus Torvalds, are available at **`http://www.kernel.org`**
  - These versions follow the development model of the kernel
  - However, they may not contain the latest development from a specific area yet. Some features in development might not be ready for mainline inclusion yet
- Many chip vendors supply their own kernel sources
- Many kernel sub-communities maintain their own kernel, with usually newer but less stable features

Getting Linux sources

- The kernel sources are available from **`http://kernel.org/pub/linux/kernel`** as **full tarballs** (complete kernel sources) and **patches** (differences between two kernel versions).
- However, more and more people use the **`git`** version control system. Absolutely needed for kernel development!
  - Fetch the entire kernel sources and history
    `git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git` (21 minutes)
  - Create a branch that starts at a specific stable version
    `git checkout -b <name-of-branch> v4.1`
  - Web interface available at
    `http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/`

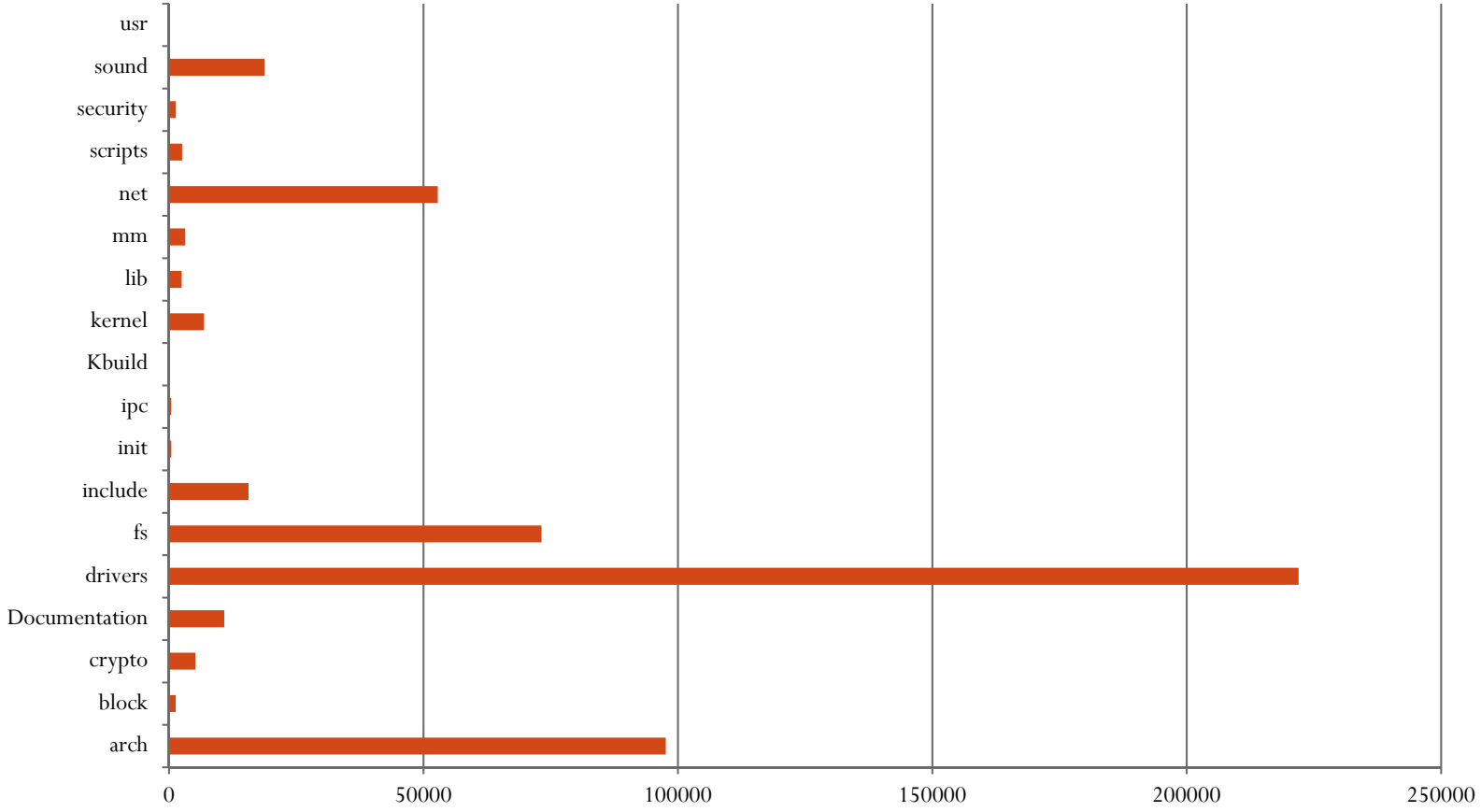
The Robert C Nelson BBB Kernel

- http://eewiki.net/display/linuxonarm/BeagleBone+Black
- **`git clone git://github.com/RobertCNelson/bb-kernel.git`**
- `host$` **`cd bb-kernel`**
- `host$` **`git checkout checkout am33x-v4.4`**
- `host$` **`./build_kernel.sh`**

Linux kernel size (1)

- Linux 3.10 sources:
  Raw size: 573 MB (43,000 files, ~15,800,000 lines)
  `gzip` compressed tar archive: 105 MB
  `bzip2` compressed tar archive: 83 MB (better)
  `xz` compressed tar archive: 69 MB (best)
- Minimum Linux 3.8.13 compiled kernel size: 5.4M
- Why are these sources so big?
  Because they include thousands of device drivers, many network protocols, support many architectures and filesystems…
- The Linux core (scheduler, memory management…) is pretty small!

Linux kernel size (3)



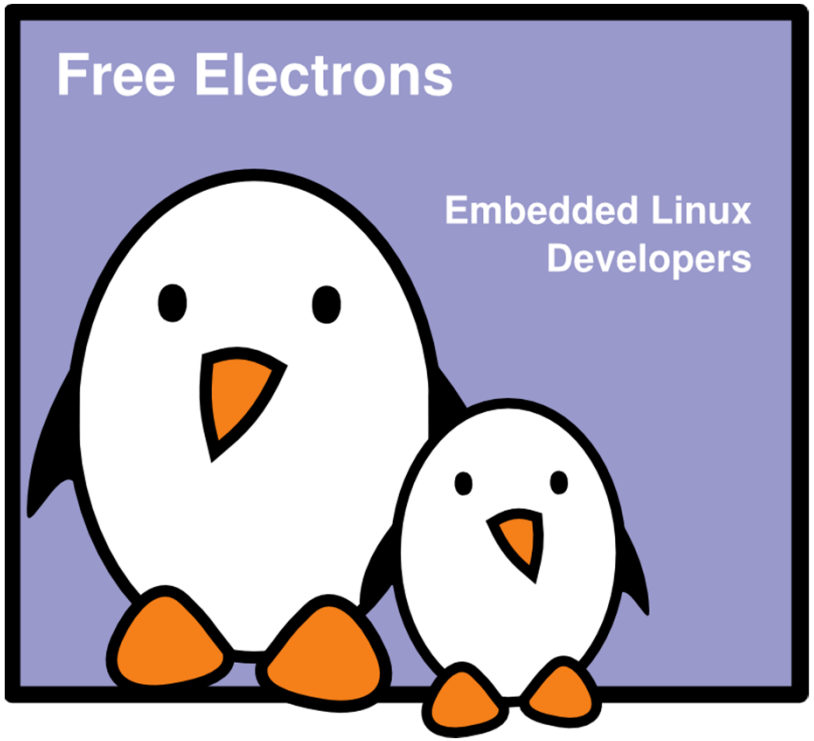
Linux 2.6.29-r46

Measured with:

du -s --apparent-size

Kernel Source Code

Michael Opdenacker
Thomas Petazzoni
Free Electrons



Latest update: 9/28/2017,
Document sources, updates and translations:
http://free-electrons.com/docs/kernel-usage
Corrections, suggestions, contributions and translations are welcome!

No C library

- The kernel has to be standalone and can't use user space code
- User space is implemented on top of kernel services, not the opposite
- Kernel code has to supply its own library implementations (string utilities, cryptography, uncompression …)
- So, you can't use standard C library functions in kernel code (`printf()`, `memset()`, `malloc()`,…).
- Fortunately, the kernel provides similar C functions for your convenience, like `printk()`, `memset()`, `kmalloc()`, …

Kernel memory constraints

- No memory protection
- Accessing illegal memory locations result in (often fatal) kernel oopses
- Fixed size stack (8 or 4 KB). Unlike in user space, there's no way to make it grow
- Kernel memory can't be swapped out (for the same reasons)

Kernel Source Code

```
host$ cd ~/BeagleBoard/bb-kernel/KERNEL
host$ ls -F
arch/             ipc/              README
block/            Kbuild            REPORTING-BUGS
certs/            Kconfig           samples/
COPYING           kernel/           scripts/
CREDITS           lib/              security/
crypto/           MAINTAINERS       sound/
Documentation/    Makefile          System.map
drivers/          mm/               tools/
firmware/         modules.builtin   usr/
fs/               modules.order     virt/
include/          Module.symvers    vmlinux*
init/             net/              vmlinux.o
```

Linux sources structure 1/5

- `arch/<ARCH>`
  - Architecture specific code
  - `arch/<ARCH>/mach-<machine>`, machine/board specific code
  - `arch/<ARCH>/include/asm`, architecture-specific headers
  - `arch/<ARCH>/boot/dts`, Device Tree source files, for some architectures
- `block/`
  - Block layer core
- `COPYING`
  - Linux copying conditions (GNU GPL)
- `CREDITS`
  - Linux main contributors
- `crypto/`
  - Cryptographic libraries

Linux sources structure 2/5

- `Documentation/`
  - Kernel documentation. Don't miss it!
- `drivers/`
  - All device drivers except sound ones (usb, pci…)
- `firmware/`
  - Legacy: firmware images extracted from old drivers
- `fs/`
  - Filesystems (fs/ext3/, etc.)
- `include/`
  - Kernel headers
- `include/linux/`
  - Linux kernel core headers
- `include/uapi/`
  - User space API headers
- `init/`
  - Linux initialization (including main.c)
- `ipc/`
  - Code used for process communication

Linux sources structure 3/5

- `Kbuild`
  - Part of the kernel build system
- `Kconfig`
  - Top level description file for configuration parameters
- `kernel/`
  - Linux kernel core (very small!)
- `lib/`
  - Misc library routines (zlib, crc32…)
- `MAINTAINERS`
  - Maintainers of each kernel part. Very useful!
- `Makefile`
  - Top Linux Makele (sets arch and version)
- `mm/`
  - Memory management code (small too!)

Linux sources structure 4/5

- `net/`
  - Network support code (not drivers)
- `README`
  - Overview and building instructions
- `REPORTING-BUGS`
  - Bug report instructions
- `samples/`
  - Sample code (markers, kprobes, kobjects…)
- `scripts/`
  - Scripts for internal or external use
- `security/`
  - Security model implementations (SELinux…)
- `sound/`
  - Sound support code and drivers
- `tools/`
  - Code for various user space tools (mostly C)

Linux sources structure 5/5

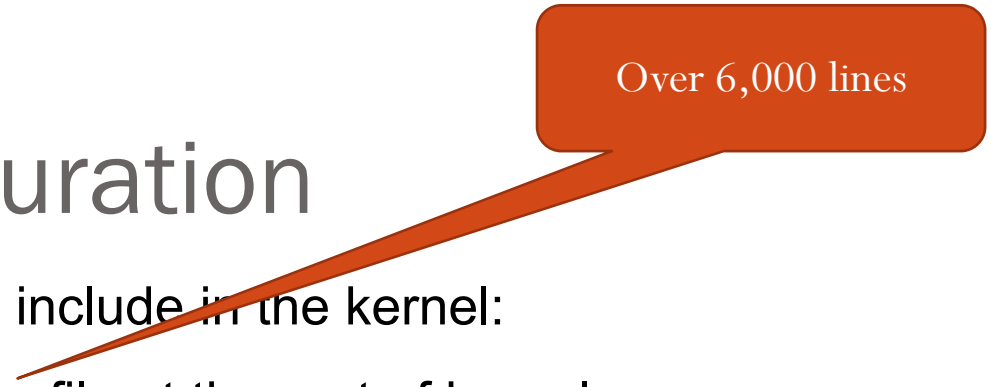
- `usr/`
  - Code to generate an initramfs cpio archive
- `virt/`
  - Virtualization support (KVM)

Embedded Linux usage

Compiling and booting Linux

Kernel configuration

Kernel configuration



Defines what features to include in the kernel:

- Stored in the **`.config`** file at the root of kernel sources.
  - Simple text file
- Most useful commands to create this config file:
  `make [xconfig|gconfig|menuconfig|oldconfig]`
- To modify a kernel in a GNU/Linux distribution:
  the configuration files are usually released in /boot/, together with kernel images: /boot/config-3.8.13-bone64
- bone$ **`ls -F /boot`**

```
config-4.4.19-ti-r41      initrd.img-4.4.21-ti-r47  uboot/
config-4.4.21-ti-r47      SOC.sh                    uEnv.txt
dtbs/                     System.map-4.4.19-ti-r41  vmlinuz-4.4.19-ti-r41*
initrd.img-4.4.19-ti-r41  System.map-4.4.21-ti-r47  vmlinuz-4.4.21-ti-r47*
```

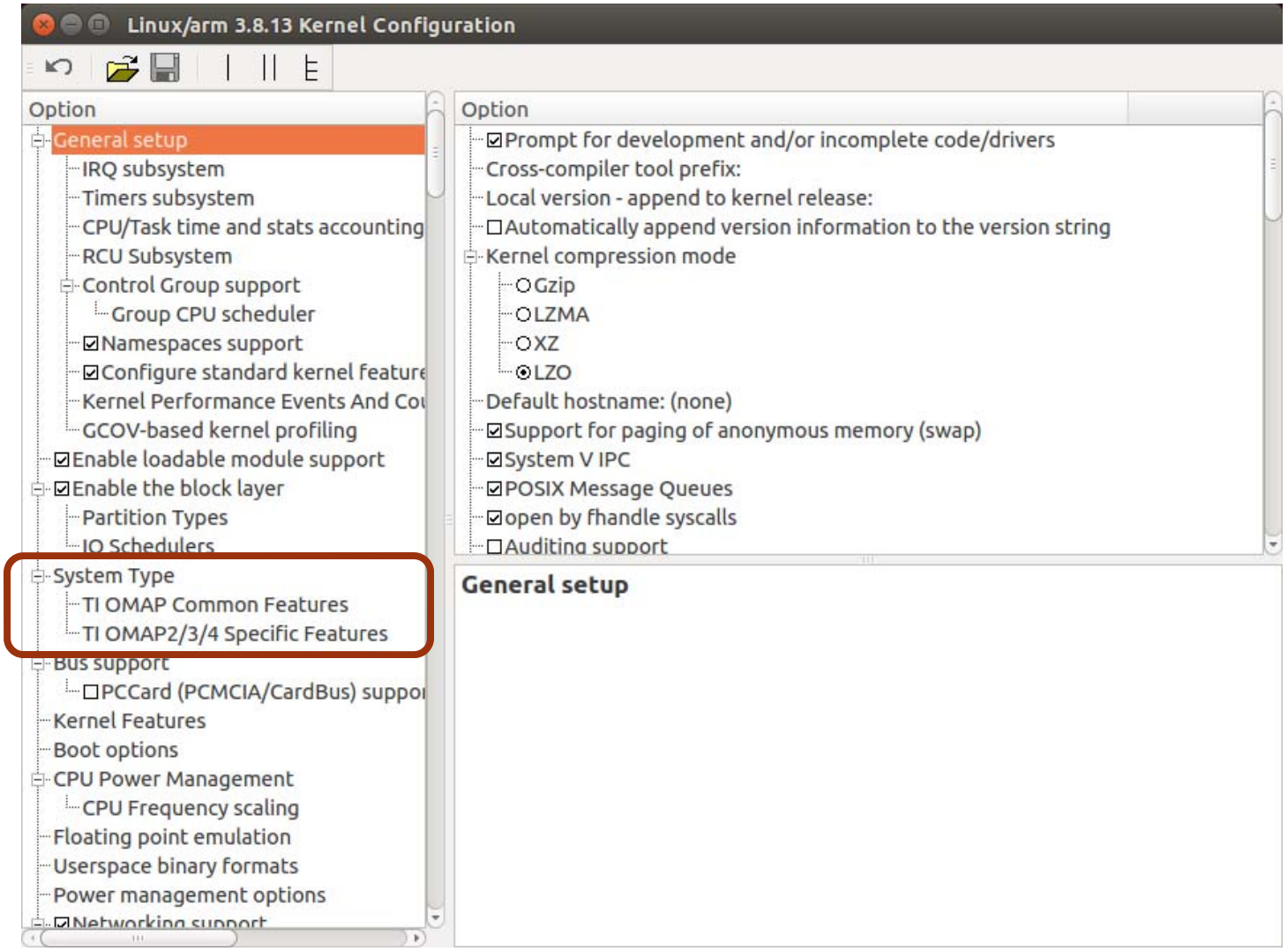
make xconfig

`make xconfig`

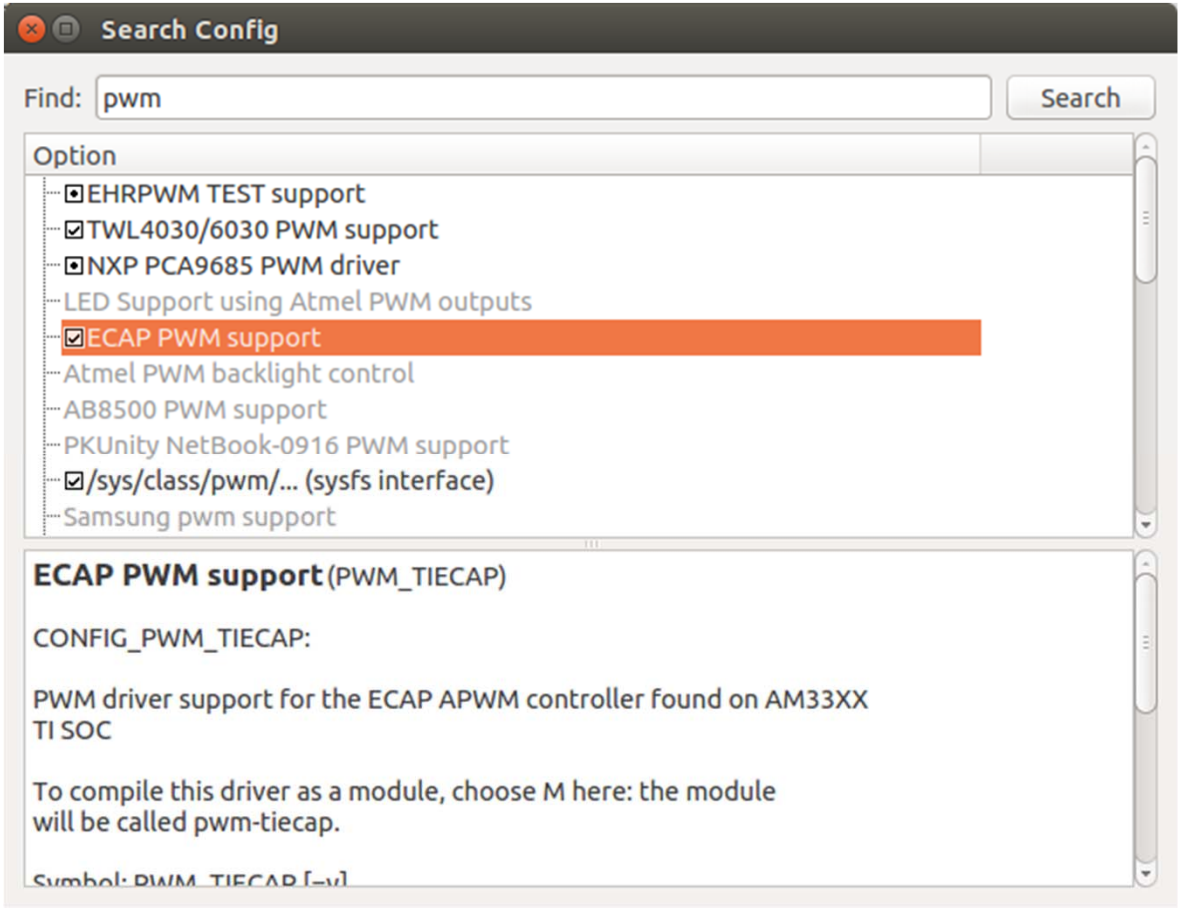
- The most common graphical interface to configure the kernel
- Make sure you read help -> introduction: useful options!
- File browser: easier to load configuration files
- New search interface to look for parameters
- Required Debian / Ubuntu packages:

```
host$ sudo apt update
host$ sudo apt install libqt4-dev
```

make xconfig screenshot



make xconfig search interface



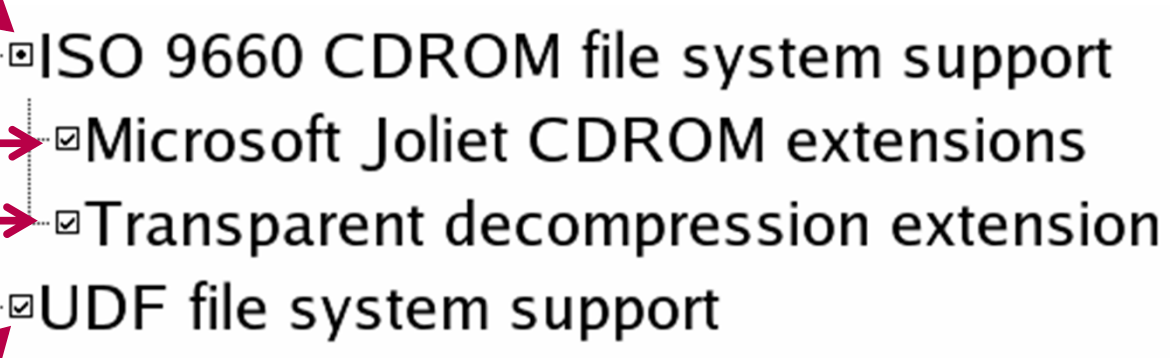
Looks for a keyword in the description string

Allows to select or unselect found parameters.

Kernel configuration options

Compiled as a module (separate file)
CONFIG_ISO9660_FS=m

Driver options
CONFIG_JOLIET=y
CONFIG_ZISOFS=y



Compiled statically into the kernel
CONFIG_UDF_FS=y

Corresponding .config file excerpt

```
#
# CD-ROM/DVD Filesystems
#
CONFIG_ISO9660_FS=m
CONFIG_JOLIET=y
CONFIG_ZISOFS=y
CONFIG_UDF_FS=y
CONFIG_UDF_NLS=y

#
# DOS/FAT/NT Filesystems
#
# CONFIG_MSDOS_FS is not set
# CONFIG_VFAT_FS is not set
CONFIG_NTFS_FS=m
# CONFIG_NTFS_DEBUG is not set
CONFIG_NTFS_RW=y
```

Section name
(helps to locate settings in the interface)

All parameters are prefixed
with CONFIG_

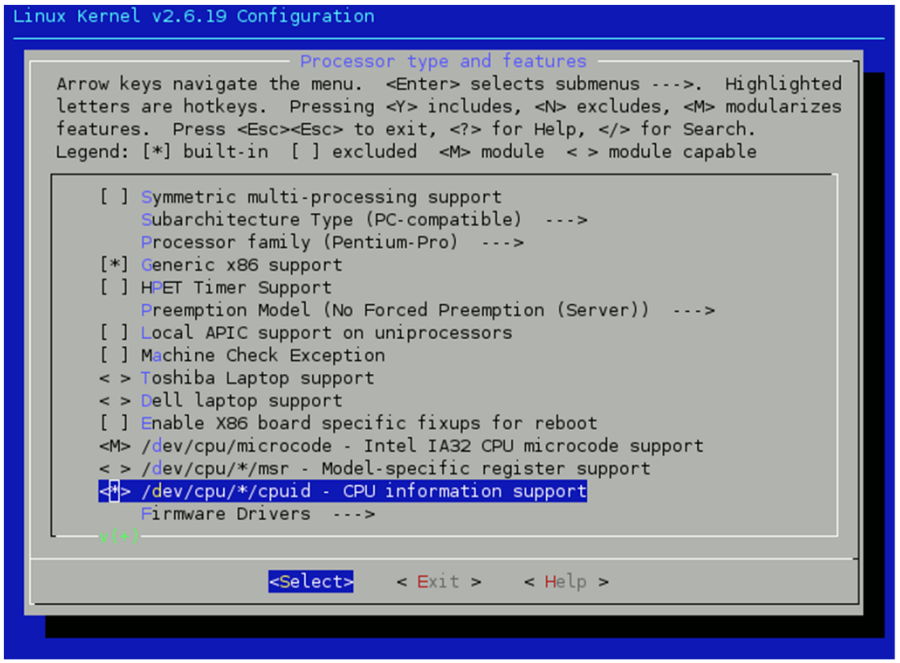
make menuconfig

make menuconfig

Useful when no graphics are available. Pretty convenient too!

Same interface found in other tools: BusyBox, buildroot...

Required Debian packages: libncurses-dev



Undoing configuration changes

A frequent problem:

- After changing several kernel configuration settings, your kernel no longer works.
- If you don't remember all the changes you made, you can get back to your previous configuration:
  > cp .config.old .config
- All the configuration interfaces of the kernel (xconfig, menuconfig, allnoconfig...) keep this .config.old backup copy.

```
host$ git diff .config
host$ git checkout .config
```

make help

make help

- Lists all available make targets
- Useful to get a reminder, or to look for new or advanced options!

Make help

```
make help
Cleaning targets:
  clean                                         - Remove most generated files but keep the config and
                    enough build support to build external modules
  mrproper                - Remove all generated files + config + various backup files
  distclean               - mrproper + remove editor backup and patch files

Configuration targets:
  config                  - Update current config utilising a line-oriented program
  nconfig         - Update current config utilising a ncurses menu based program
  menuconfig              - Update current config utilising a menu based program
  xconfig                 - Update current config utilising a QT based front-end
  gconfig                 - Update current config utilising a GTK based front-end
  oldconfig               - Update current config utilising a provided .config as base
  localmodconfig  - Update current config disabling modules not loaded
  localyesconfig  - Update current config converting local mods to core
  silentoldconfig - Same as oldconfig, but quietly, additionally update deps
  defconfig               - New config with default from ARCH supplied defconfig
  savedefconfig   - Save current config as ./defconfig (minimal config)
  allnoconfig             - New config where all options are answered with no
  allyesconfig            - New config where all options are accepted with yes
  allmodconfig            - New config selecting modules when possible
  alldefconfig    - New config with all symbols set to default
  randconfig              - New config with random answer to all options
  listnewconfig   - List new options
  oldnoconfig     - Same as silentoldconfig but set new symbols to n (unset)

Other generic targets:
  all                                           - Build all targets marked with [*]
* vmlinux                 - Build the bare kernel
* modules                 - Build all modules
  modules_install - Install all modules to INSTALL_MOD_PATH (default: /)
  firmware_install- Install all firmware to INSTALL_FW_PATH
                    (default: $(INSTALL_MOD_PATH)/lib/firmware)
  dir/            - Build all files in dir and below
  dir/file.[oisS] - Build specified target only
```

Make help

```
Configuration targets:
  config       - Update current config utilising a line-oriented program
  nconfig      - Update current config utilising a ncurses menu based program
  menuconfig   - Update current config utilising a menu based program
  xconfig      - Update current config utilising a QT based front-end
  gconfig      - Update current config utilising a GTK based front-end
  oldconfig    - Update current config utilising a provided .config as base
  localmodconfig  - Update current config disabling modules not loaded
  localyesconfig  - Update current config converting local mods to core
  silentoldconfig - Same as oldconfig, but quietly, additionally update deps
  defconfig       - New config with default from ARCH supplied defconfig
  savedefconfig   - Save current config as ./defconfig (minimal config)
  allnoconfig     - New config where all options are answered with no
  allyesconfig    - New config where all options are accepted with yes
  allmodconfig    - New config selecting modules when possible
  alldefconfig    - New config with all symbols set to default
  randconfig      - New config with random answer to all options
  listnewconfig   - List new options
  oldnoconfig     - Same as silentoldconfig but set new symbols to n (unset)
```

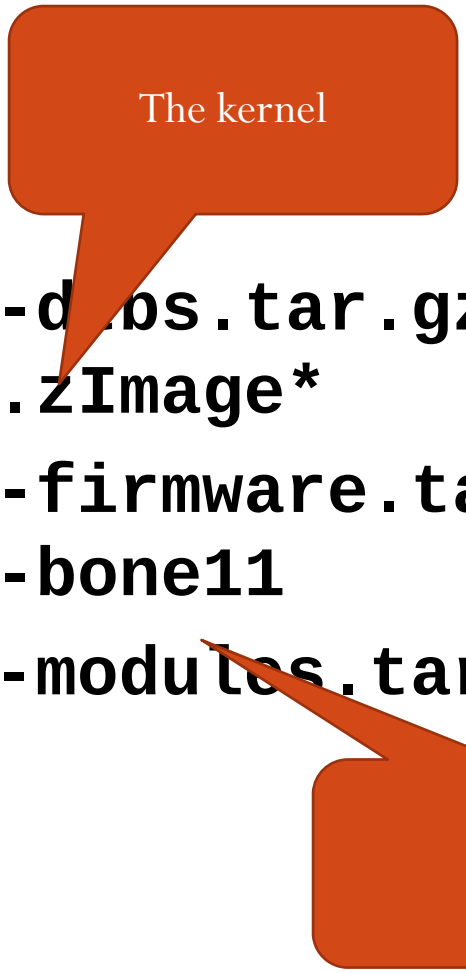
Embedded Linux usage

Compiling and installing the kernel
for the host system

Installing a new kernel

- When using Nelson's tools a new kernel is put in **bb-kernel/deploy**

```
host$ ls -sh
total 67M
328K 4.4.15-bone11-dtbs.tar.gz
7.1M 4.4.15-bone11.zImage*
1.2M 4.4.15-bone11-firmware.tar.gz
140K config-4.4.15-bone11
 58M 4.4.15-bone11-modules.tar.gz
```



Installing

- First load sshfs

```
host$ sudo apt install sshfs
```

- Then copy **may_install_kernel.sh** to the **bb-kernel** directory

```
host$ cd ~/BeagleBoard/bb-kernel
host$ cp ~/BeagleBoard/exercises/linux/kernel/may_install_kernel.sh tools
host$ tools/may_install_kernel.sh
```

- Note, the command must be run from **bb-kernel**, not the tools directory.
- The script will mount the Bone's root file system in **bb-kernel/deploy/disk** and then copy the needed files to it. Once done you can reboot your bone. If you are done with the mounted files you can unmout them with

```
host$ sudo umount deploy/disk
```

Compiling and installing the kernel

Compiling step

- make

You can speed up compiling by running multiple compile jobs in parallel, especially if you have multiple CPU cores.

Example: make -j 4

MAY1

MAY1 How do you build for the target?

Mark A. Yoder, 12/22/2009

Kernel cleanup targets

- Clean-up generated files
  (to force re-compiling drivers):
  make clean
- Remove all generated files. Needed when switching from one architecture to another
  Caution: also removes your .config file!
  make mrproper
- Also remove editor backup and patch reject files:
  (mainly to generate patches):
  make distclean

Generated files

Created when you run the make command. The kernel is in fact a single binary image, nothing more !

- …/vmlinux
  Raw Linux kernel image, non compressed.
- …/arch/<arch>/boot/zImage (default image on arm)
  zlib compressed kernel image
- …/arch/<arch>/boot/bzImage (default image on x86)
  Also a zlib compressed kernel image.
  Caution: bz means “big zipped” but not “bzip2 compressed”!

News: new compression formats are now available since 2.6.30: lzma and bzip2. Free Electrons also contributed lzo support (very fast decompression).

Files created by make install

- /boot/vmlinuz-<version>
  Compressed kernel image. Same as the one in /arch/<arch>/boot
- /boot/System.map-<version>
  Stores kernel symbol addresses
- /boot/config-<version>
  Kernel configuration for this version

Files created by make modules_install

/lib/modules/<version>/: Kernel modules + extras

- kernel/
  Module .ko (Kernel Object) files, in the same directory structure as in the sources.
- modules.alias
  Module aliases for module loading utilities. Example line:
  alias sound-service-?-0 snd_mixer_oss
- modules.dep
  Module dependencies
- modules.symbols
  Tells which module a given symbol belongs to.

All the files in this directory are text files.

Don't hesitate to have a look by yourself!