08-1 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 –

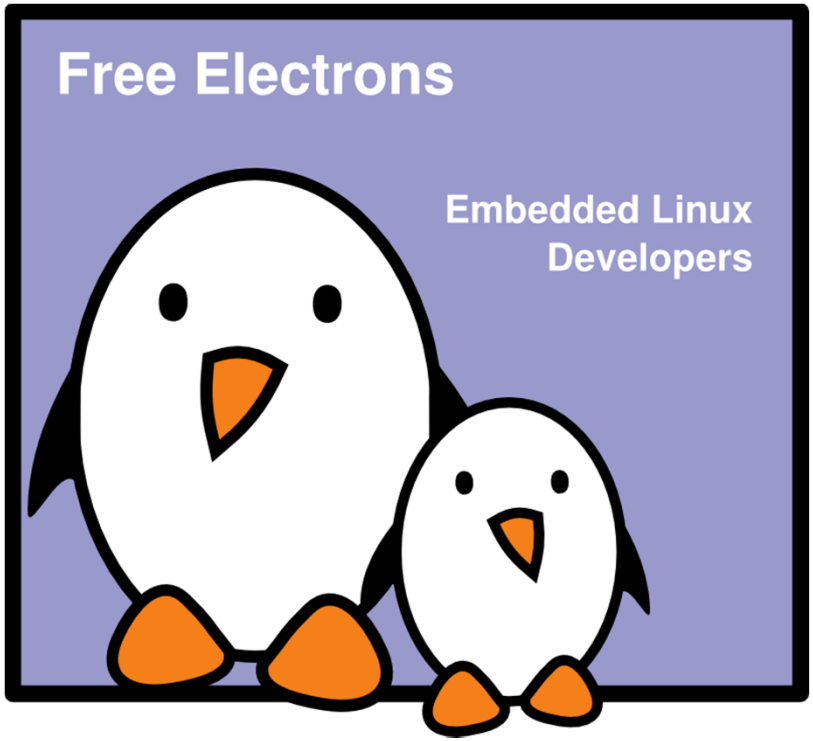
I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat(same physical layout of the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

Linux kernel introduction

Michael Opdenacker
Thomas Petazzoni
Free Electrons



Latest update: 9/28/2015,
Document sources, updates and translations:
http://free-electrons.com/docs/kernel-intro
Corrections, suggestions, contributions and translations are welcome!

Embedded Linux driver development

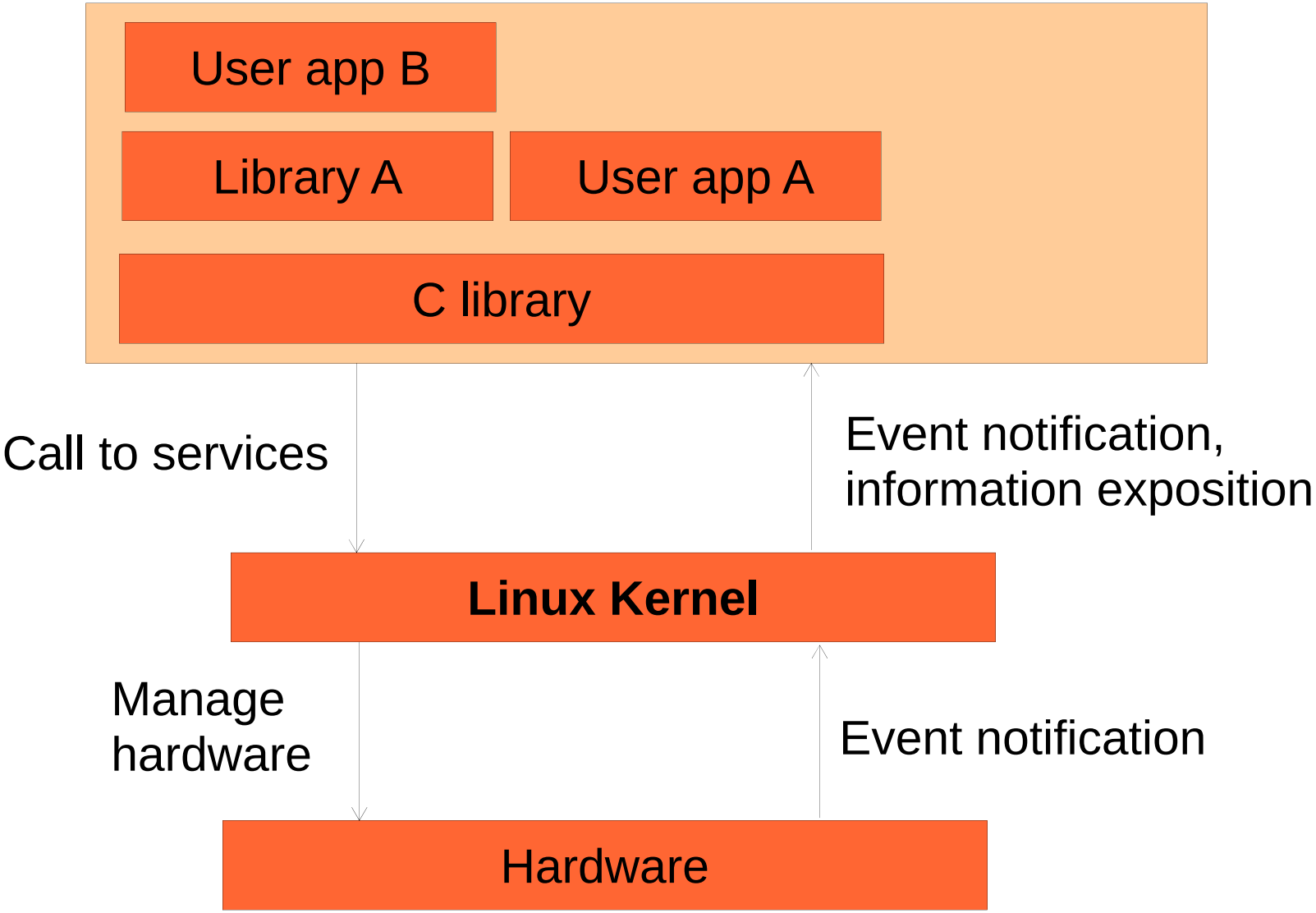
Kernel overview

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

What's the current version?

4.2.1

2.6.31 status

- 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: 4.2.1

mainline:	4.3-rc2	2015-09-20	[tar.xz]	[pgp]	[patch]		[view diff]	[browse]	
stable:	4.2.1	2015-09-21	[tar.xz]	[pgp]	[patch]		[view diff]	[browse]	[changelog]
longterm:	4.1.8	2015-09-21	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.18.21	2015-08-31	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.14.53	2015-09-21	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.12.48	2015-09-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.10.89	2015-09-21	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.4.109	2015-09-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.2.71	2015-08-12	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	2.6.32.68	2015-09-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20150923	2015-09-23						[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+

This site is operated by the Linux Kernel Organization, Inc., a 501(c)3 nonprofit corporation, with support from the following sponsors.

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 ca
  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:

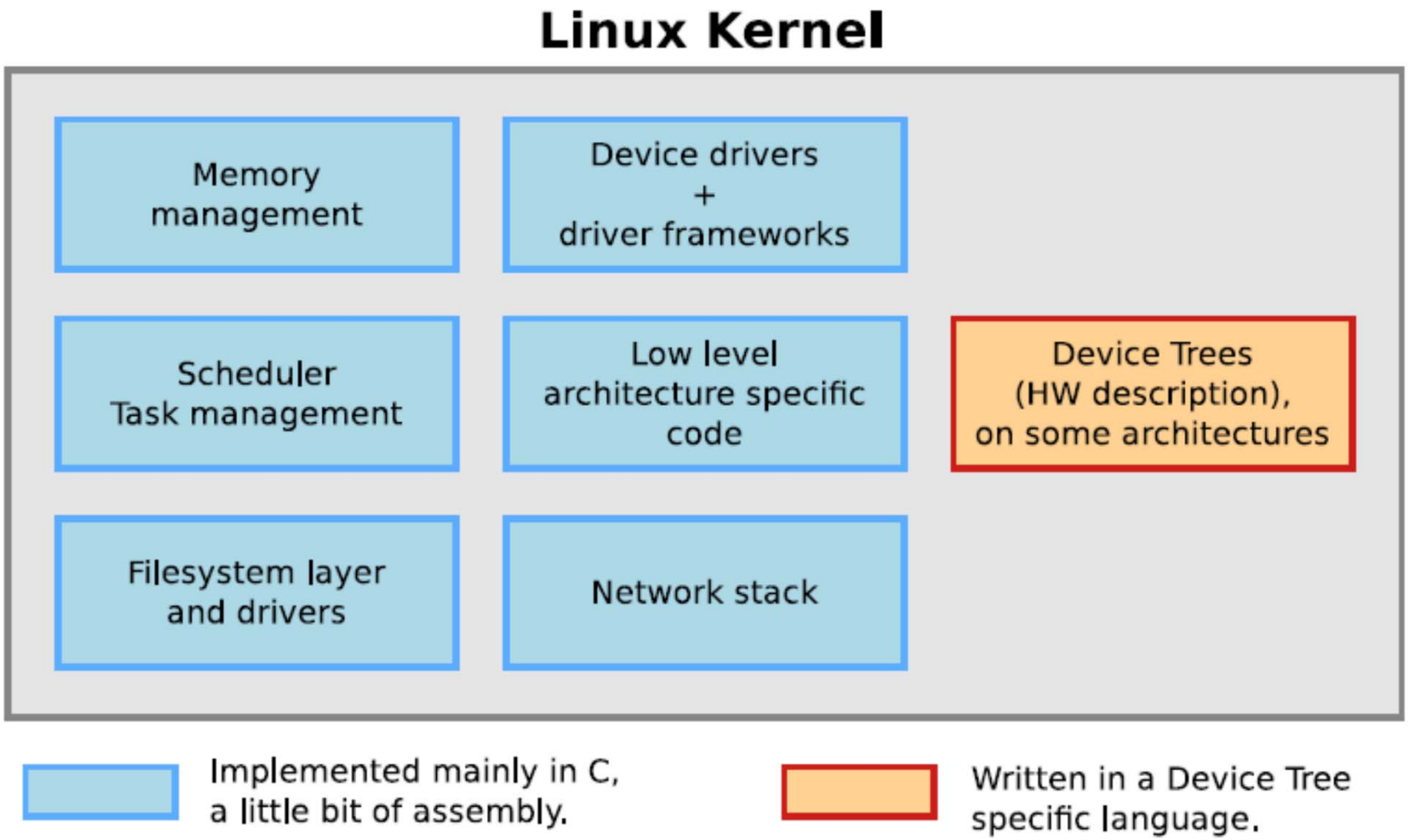
- /proc/cpuinfo: processor information
- /proc/meminfo: memory status
- /proc/version: kernel version and build information
- /proc/cmdline: kernel command line
- /proc/<pid>/environ: calling environment
- /proc/<pid>/cmdline: process command line

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/    16/   36/   45/   75/        cpuinfo       kmsg          slabinfo
10/   17/   38/   46/   76/        crypto        kpagecount    softirqs
101/  18/   39/   5/    79/        device-tree/  kpageflags    stat
11/   19/   40/   53/   8/         devices       loadavg       swaps
12/   2/    41/   530/  80/        diskstats     locks         sys/
127/  20/   412/  531/  81/        dri/          meminfo       sysrq-trigger
129/  21/   418/  533/  87/        driver/       misc          sysvipc/
13/   24/   42/   563/  88/        execdomains   modules       timer_list
138/  243/  429/  564/  9/         fb            mounts@       timer_stats
139/  244/  430/  565/  asound/    filesystems   mtd           tty/
14/   245/  437/  567/  buddyinfo  fs/           net@          uptime
140/  261/  440/  57/   bus/       interrupts    pagetypeinfo  version
142/  268/  442/  6/    cgroups    iomem         partitions    vmallocinfo
144/  27/   443/  69/   cmdline    ioports       sched_debug   vmstat
145/  3/    445/  7/    config.gz  irq/          schedstat     zoneinfo
151/  320/  447/  73/   consoles   kallsyms      scsi/
152/  345/  449/  74/   cpu/       key-users     self@
```

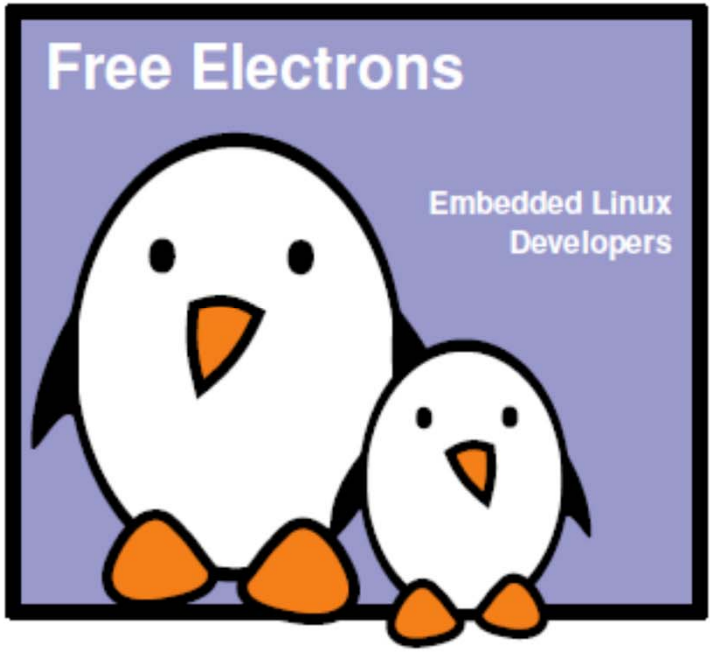
Inside the Linux kernel



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

1.1. 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

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

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 4.0 has been released on Sun, 12 Apr 2015.

Summary: This release adds support for live patching the kernel code, aimed primarily at fixing security updates without rebooting; DAX, a way to avoid using the kernel cache when filesystems run on systems with persistent memory storage; kasan, a dynamic memory error detector that allows to find use-after-free and out-of-bounds bugs; lazytime, an alternative to relatime, which causes access, modified and changed time updates to only be made in the cache and written to the disk opportunistically; allow overlayfs to have multiple lower layers, support of Parallel NFS server architecture; and dm-crypt CPU scalability improvements. There are also new drivers and many other small improvements.

1. Prominent features

1.1. Arbitrary version change

This release increases the version to 4.0. This switch from 3.x to 4.0 version numbers is, however, entirely meaningless and it should not be associated to any important changes in the kernel. This release could have been 3.20, but Linus Torvalds just got tired of the old number, made a poll, and changed it. Yes, it is frivolous. The less you think about it, the better.

1.2. Live patching

This release introduces "livepatch", a feature for live patching the kernel code, aimed primarily at systems who want to get security updates without needing to reboot. This feature has been born as result of merging kgraft and kpatch, two attempts by SuSE and Red Hat that where started to replace the now propietary

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> v3.11`
  - Web interface available at
    `http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/`

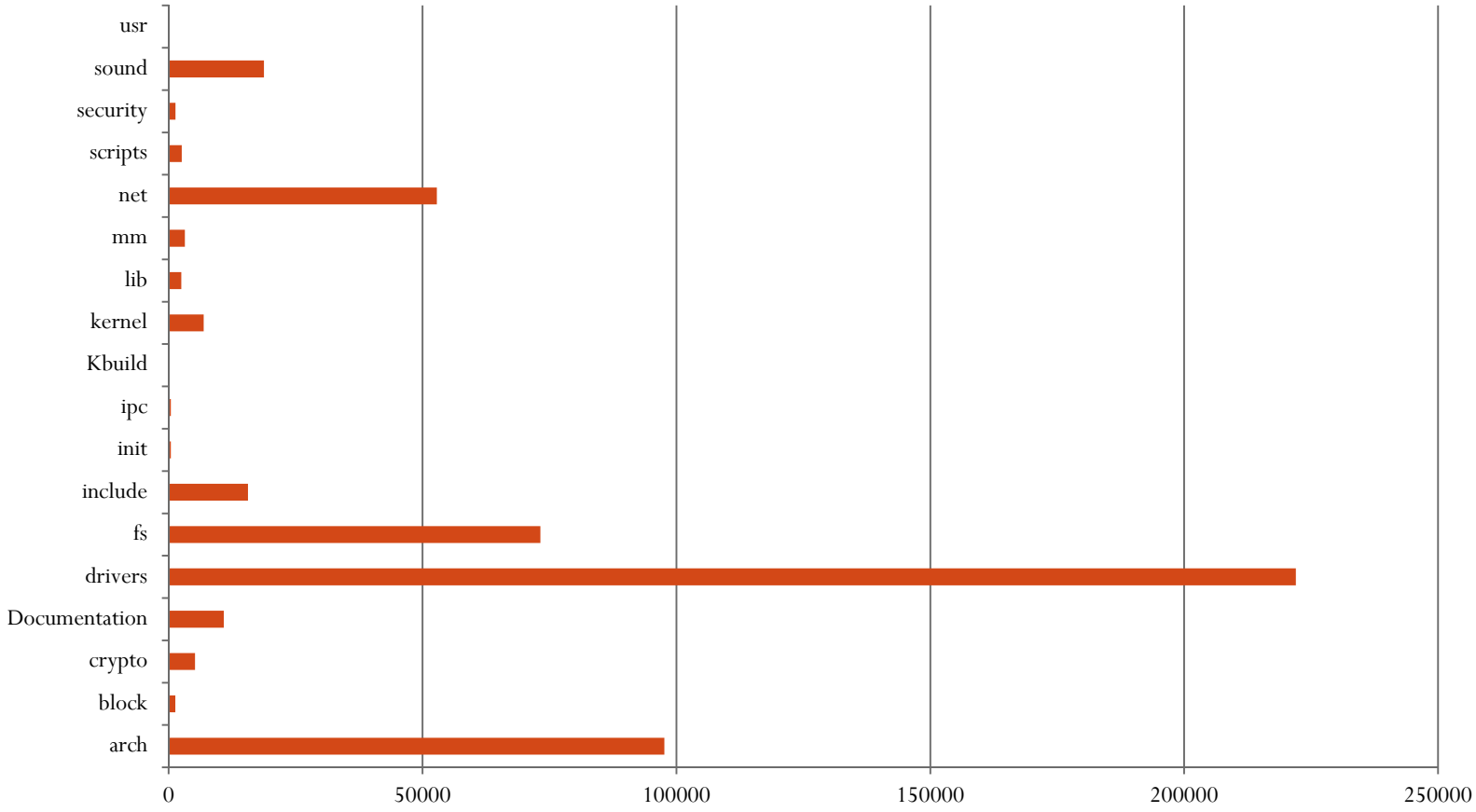
The Robert C Nelson BBB Kernel

- http://eewiki.net/display/linuxonarm/BeagleBone+Black
- `git clone git://github.com/RobertCNelson/bb-kernel.git`
- host$ **cd bb-kernel**
- host$ **git tag** (This shows what versions can be checked out.)
- host$ **git checkout 3.8.13-bone77 -b 3.8.13-bone77**
- 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)



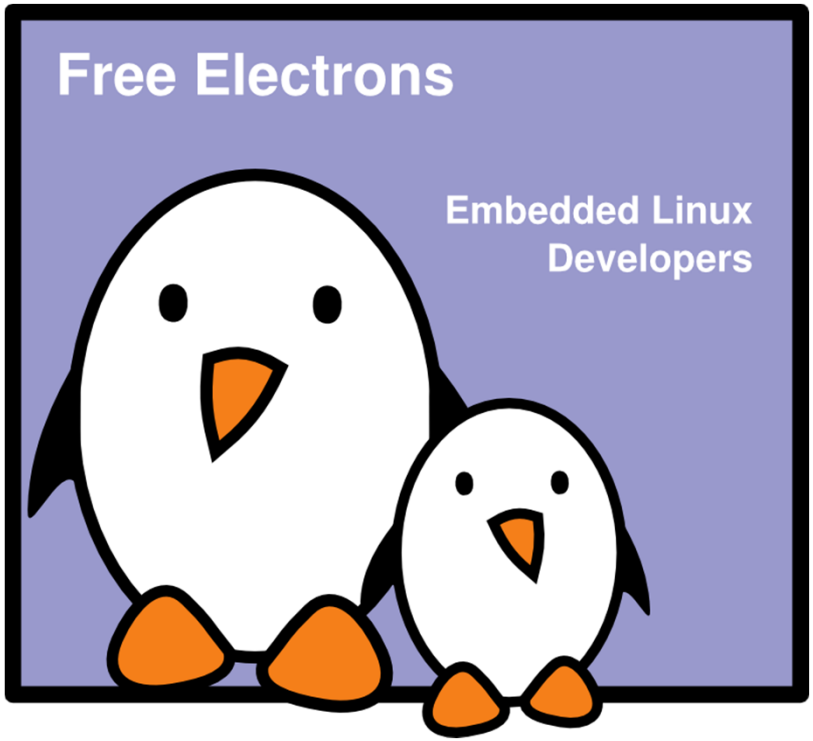
Linux 2.6.29-r46

Measured with:

du -s --apparent-size

Kernel Source Code

Michael Opdenacker
Thomas Petazzoni
Free Electrons



Latest update: 9/28/2015,
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/              Kbuild             REPORTING-BUGS
block/             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
ipc/               README
```

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

Nearly 5,000 lines

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
- `beagle$ ls -F /boot`

```
config-3.14.17-bone8      initrd.img-3.8.13-bone64  uboot/
config-3.8.13-bone64      SOC.sh                    uEnv.txt
dtbs/                     System.map-3.14.17-bone8  vmlinuz-3.14.17-bone8*
initrd.img-3.14.17-bone8  System.map-3.8.13-bone64  vmlinuz-3.8.13-bone64*
```

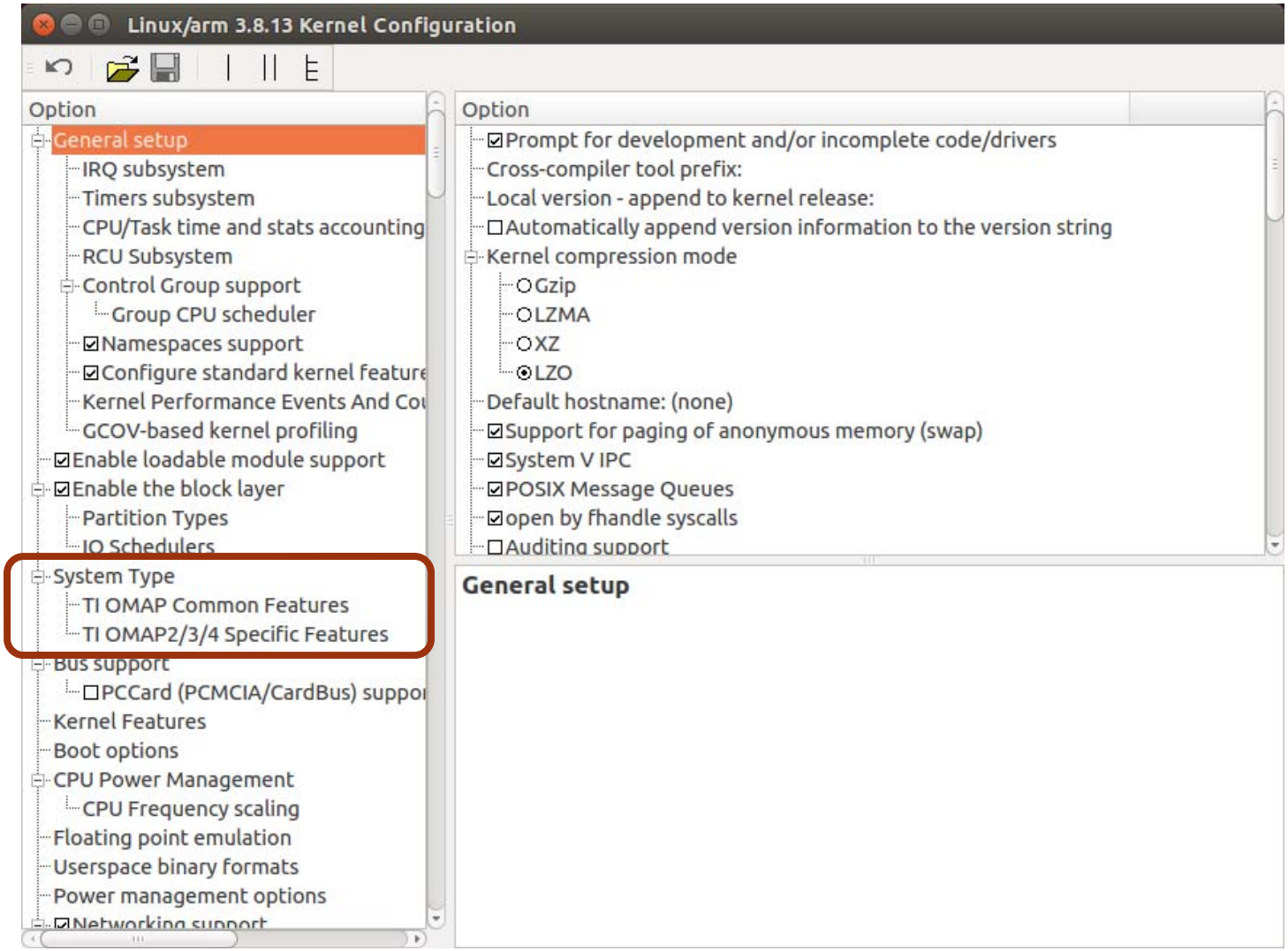
make xconfig

`make xconfig`

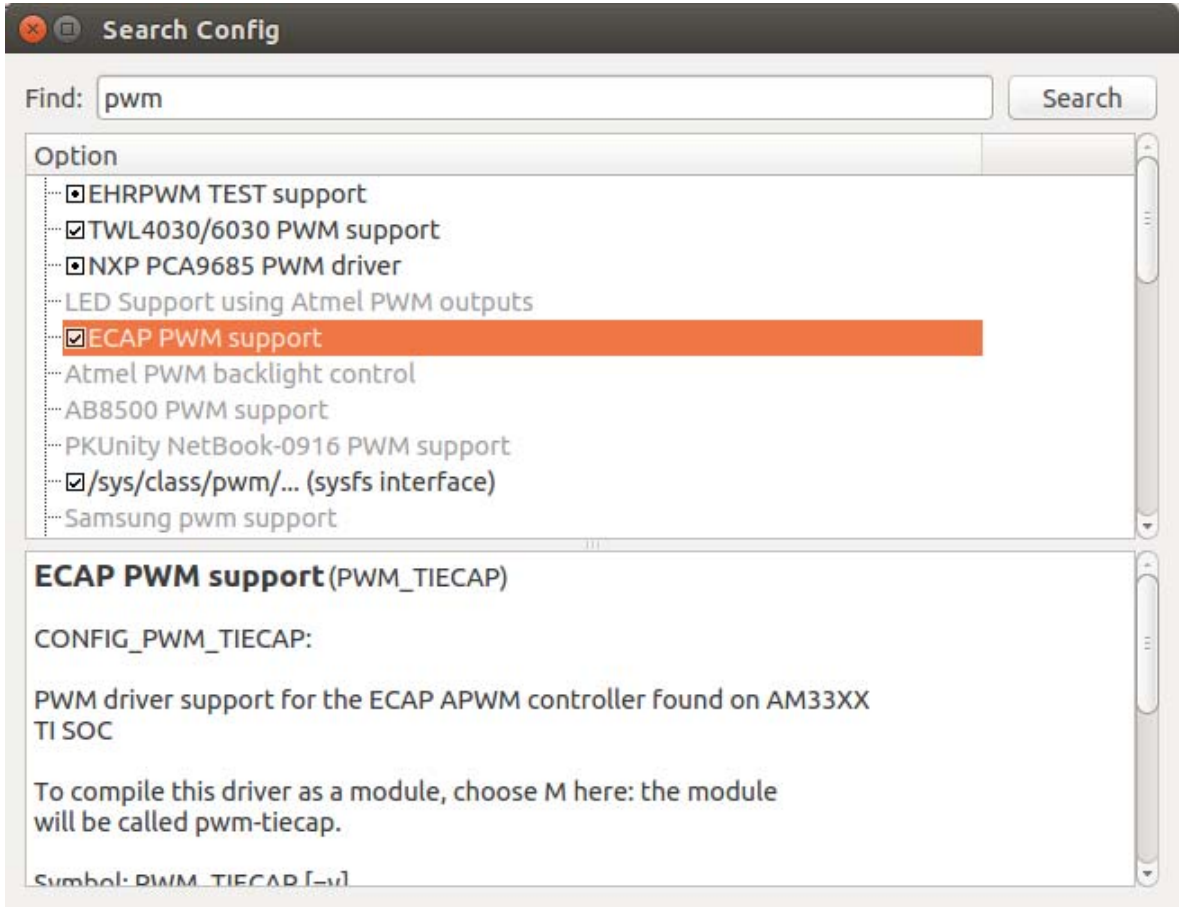
- The most common graphical interface to configure the kernel
- Make sure you read
  help -> introduction: useful options!
- File browser: easier to load configuration files
- New search interface to look for parameters
- Required Debian / Ubuntu packages:

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

make xconfig screenshot



make xconfig search interface



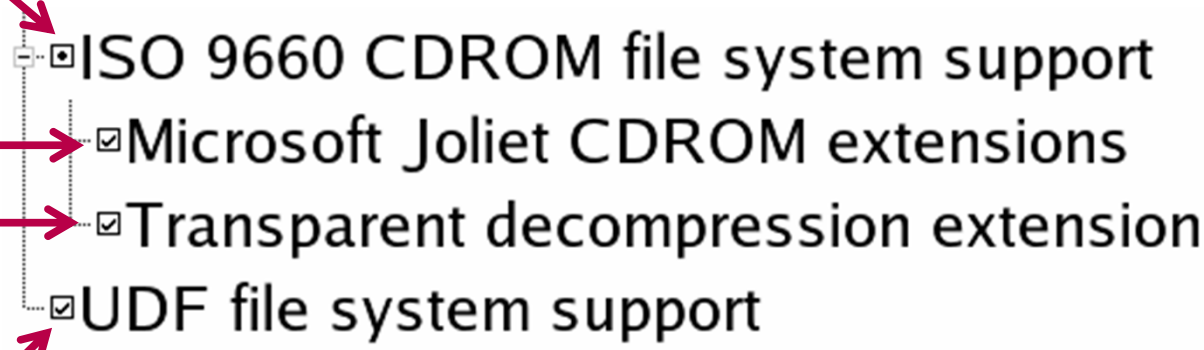
Looks for a keyword in the description string

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_

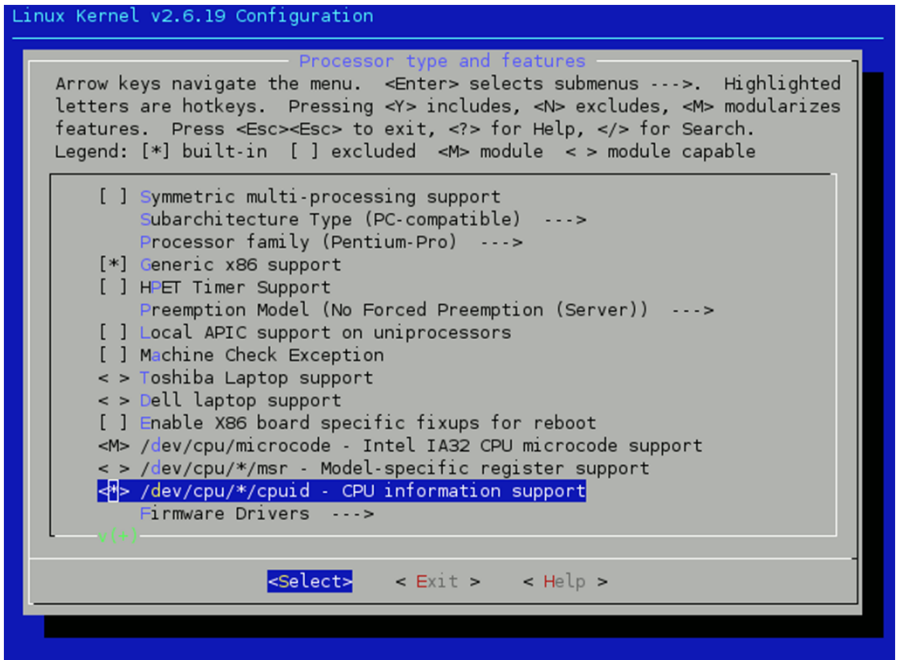
make menuconfig

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.[ois]  - 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 19M
280K 3.8.13-bone77-dtbs.tar.gz
5.3M 3.8.13-bone77.zImage
1.2M 3.8.13-bone77-firmware.tar.gz
108K config-3.8.13-bone77
 12M 3.8.13-bone77-modules.tar.gz
```

The kernel

.config

Installing

- First load sshfs

```
host$ sudo apt-get install sshfs
```

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

```
host$ cd ~/BeagleBoard/bb-kernel
host$ cp ~/BeagleBoard/exercises/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
```