06-1 The Kernel

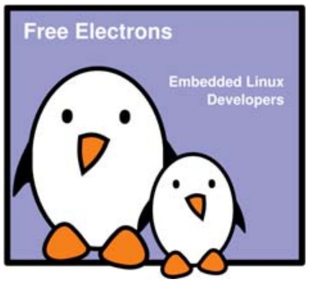
It all started with…

```
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



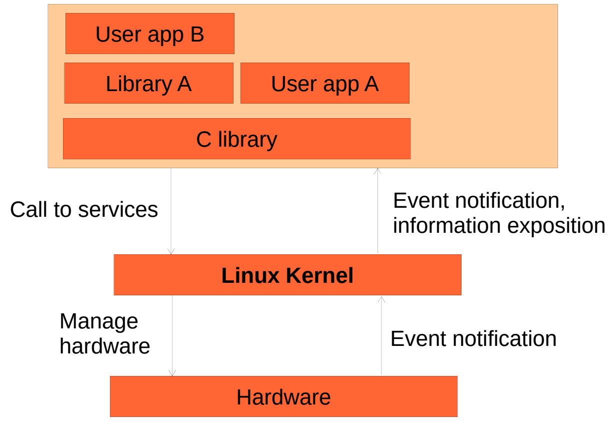
Latest update: 1/16/2012,
Document sources, updates and translations:
http://free-electrons.com/docs/kernel-intro
Corrections, suggestions, contributions and translations are welcome!

Embedded Linux driver development

Kernel overview

Linux features

Linux kernel in the system



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 key features

- Portability and hardware support. Runs on most architectures.
- Scalability
  Can run on super computers as well as on tiny devices
  (4 MB of RAM is enough).
- Compliance to standards and interoperability.
- Exhaustive networking support.
- Security
  It can't hide its flaws. Its code is reviewed by many experts.
- Stability and reliability.
- Modularity
  Can include only what a system needs even at run time.
- Easy to program
  You can learn from existing code. Many useful resources on the net.

Supported hardware architectures

2.6.31 status

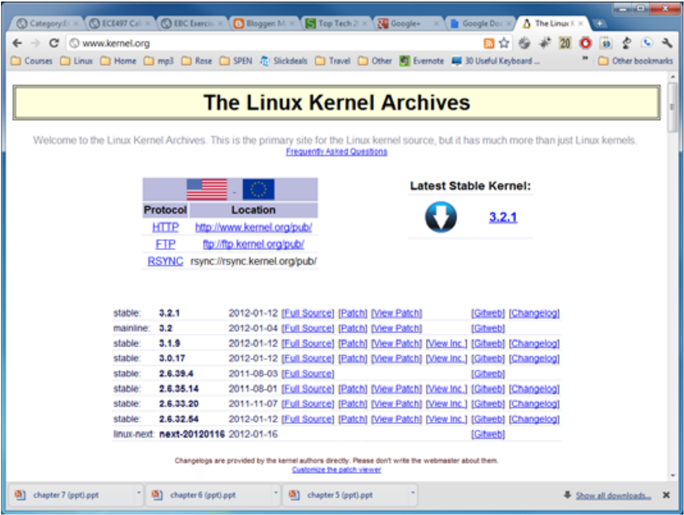
What's the current version?

3.2.1

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

How did I find it?

kernel.org



www.kernel.org

The Linux Kernel Archives

Welcome to the Linux Kernel Archives. This is the primary site for the Linux kernel source, but it has much more than just Linux kernels.
Frequently Asked Questions

Protocol	Location
HTTP	http://www.kernel.org/pub/
FTP	ftp://ftp.kernel.org/pub/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel:

3.2.1

stable:	**3.2.1**	2012-01-12	[Full Source]	[Patch]	[View Patch]		[Gitweb]	[Changelog]
mainline:	**3.2**	2012-01-04	[Full Source]	[Patch]	[View Patch]		[Gitweb]	
stable:	**3.1.9**	2012-01-12	[Full Source]	[Patch]	[View Patch]	[View Inc.]	[Gitweb]	[Changelog]
stable:	**3.0.17**	2012-01-12	[Full Source]	[Patch]	[View Patch]	[View Inc.]	[Gitweb]	[Changelog]
stable:	**2.6.39.4**	2011-08-03	[Full Source]				[Gitweb]	
stable:	**2.6.35.14**	2011-08-01	[Full Source]	[Patch]	[View Patch]	[View Inc.]	[Gitweb]	[Changelog]
stable:	**2.6.33.20**	2011-11-07	[Full Source]	[Patch]	[View Patch]	[View Inc.]	[Gitweb]	[Changelog]
stable:	**2.6.32.54**	2012-01-12	[Full Source]	[Patch]	[View Patch]	[View Inc.]	[Gitweb]	[Changelog]
linux-next:	**next-20120116**	2012-01-16					[Gitweb]	

Changelogs are provided by the kernel authors directly. Please don't write the webmaster about them.
Customize the patch viewer

System calls

- 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 stable over time: only new system calls can be added by the kernel developers
- This system call interface is wrapped by the C library, and userspace applications usually never make a system call directly but rather use the corresponding C library function

Virtual filesystems

- Linux makes system and kernel information available in user-space through virtual filesystems (virtual files not existing on any real storage). No need to know kernel programming to access such information!
- Mounting /proc:
  sudo mount -t proc none /proc
- Mounting /sys:
  sudo mount -t sysfs none /sys

Filesystem type

Raw device or filesystem image
In the case of virtual filesystems, any string is fine

Mount point

/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

... and many more! See by yourself!

Lots of details about the /proc interface are available in Documentation/filesystems/proc.txt (almost 2000 lines) in the kernel sources.

Embedded Linux usage

Kernel overview

Linux versioning scheme and development process

Changes since Linux 2.6 (1)

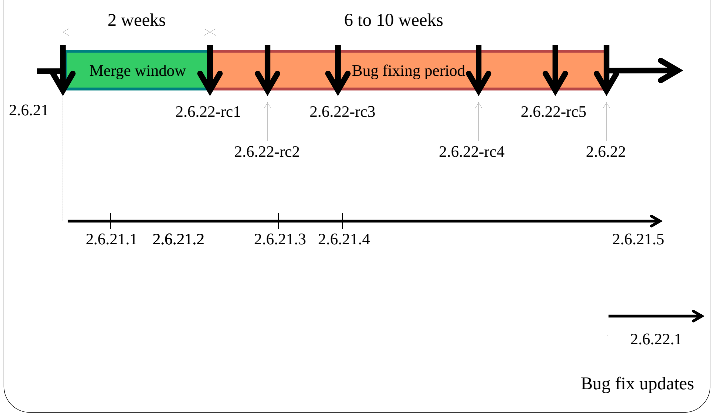
- Since 2.6.0, kernel developers have been able to introduce lots of new features one by one on a steady pace, without having to make major changes in existing subsystems.
- Opening a new Linux 2.7 (or 2.9) development branch will be required only when Linux 2.6 is no longer able to accommodate key features without undergoing traumatic changes.
- Thanks to this, more features are released to users at a faster pace.

Changes since Linux 2.6 (2)

Since 2.6.14, the kernel developers agreed on the following development model:

- After the release of a 2.6.x version, a two-weeks merge window opens, during which major additions are merged.
- The merge window is closed by the release of test version 2.6.(x+1)-rc1
- The bug fixing period opens, for 6 to 10 weeks.
- At regular intervals during the bug fixing period, 2.6.(x+1)-rcY test versions are released.
- When considered sufficiently stable, kernel 2.6.(x+1) is released, and the process starts again.

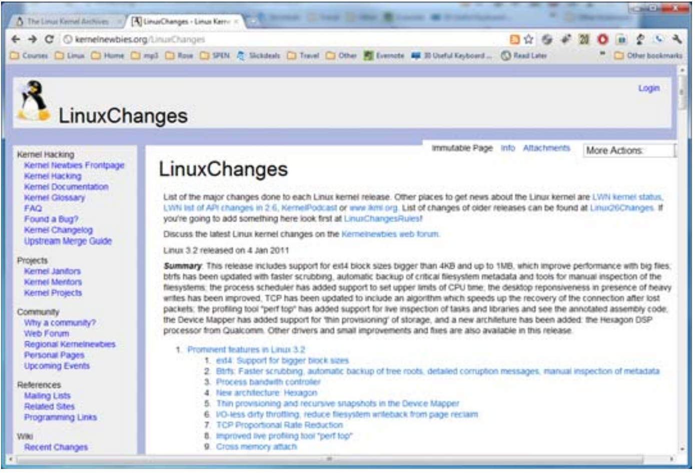
Merge and bug fixing windows



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



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

LinuxChanges

List of the major changes done to each Linux kernel release. Other places to get news about the Linux kernel are LWN kernel status, LWN list of API changes in 2.6, KernelPodcast or www.lkml.org. List of changes of older releases can be found at Linux26Changes. If you're going to add something here look first at LinuxChangesRules!

Discuss the latest Linux kernel changes on the Kernelnewbies web forum.

Linux 3.2 released on 4 Jan 2011

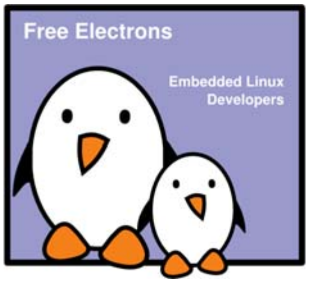
Summary: This release includes support for ext4 block sizes bigger than 4KB and up to 1MB, which improve performance with big files; btrfs has been updated with faster scrubbing, automatic backup of critical filesystem metadata and tools for manual inspection of the filesystems; the process scheduler has added support to set upper limits of CPU time; the desktop reponsiveness in presence of heavy writes has been improved, TCP has been updated to include an algorithm which speeds up the recovery of the connection after lost packets; the profiling tool "perf top" has added support for live inspection of tasks and libraries and see the annotated assembly code; the Device Mapper has added support for 'thin provisioning' of storage, and a new architeture has been added: the Hexagon DSP processor from Qualcomm. Other drivers and small improvements and fixes are also available in this release.

1. Prominent features in Linux 3.2
   1. ext4: Support for bigger block sizes
   2. Btrfs: Faster scrubbing, automatic backup of tree roots, detailed corruption messages, manual inspection of metadata
   3. Process bandwith controller
   4. New architecture: Hexagon
   5. Thin provisioning and recursive snapshots in the Device Mapper
   6. I/O-less dirty throttling, reduce filesystem writeback from page reclaim
   7. TCP Proportional Rate Reduction
   8. Improved live profiling tool "perf top"
   9. Cross memory attach

Embedded Linux kernel usage

Embedded Linux
kernel usage

Michael Opdenacker
Thomas Petazzoni
Free Electrons



Latest update: 1/16/2012,
Document sources, updates and translations:
http://free-electrons.com/docs/kernel-usage
Corrections, suggestions, contributions and translations are welcome!

Embedded Linux usage

Compiling and booting Linux
Linux kernel sources

Location of kernel sources

- The official version of the Linux kernel, as released by Linus Torvalds is available at http://www.kernel.org
  - This version follows the well-defined development model of the kernel
  - However, it may not contain the latest development from a specific area, due to the organization of the development model and because features in development might not be ready for mainline inclusion

Like omap

- Many kernel sub-communities maintain their own kernel, with usually newer but less stable features
  - Architecture communities (ARM, MIPS, PowerPC, etc.), device drivers communities (I2C, SPI, USB, PCI, network, etc.), other communities (real-time, etc.)
  - They generally don't release official versions, only development trees are available

Linux kernel size (1)

- Linux 2.6.31 sources:
  Raw size: 350 MB (30,900 files, approx 12,000,000 lines)
  gzip compressed tar archive: 75 MB
  bzip2 compressed tar archive: 59 MB (better)
  lzma compressed tar archive: 49 MB (best)
- Minimum Linux 2.6.29 compiled kernel size with CONFIG_EMBEDDED,
  for a kernel that boots a QEMU PC (IDE hard drive, ext2 filesystem, ELF executable support):
  532 KB (compressed), 1325 KB (raw)
- 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 (1)

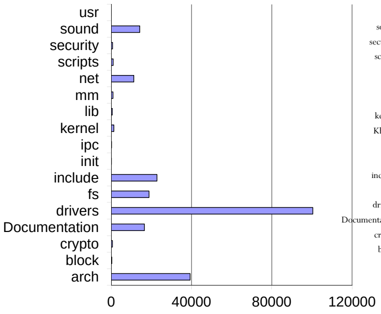
- Linux 2.6.31 sources:
  Raw size: 350 MB (30,900 files, approx 12,000,000 lines)
  gzip compressed tar archive: 75 MB
  bzip2 compressed tar archive: 59 MB (better)
  lzma compressed tar archive: 49 MB (best)
- Linux 2.6.32 sources:

  1.3G
- Linux 3.0.9 sources:

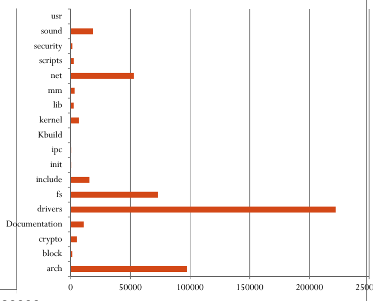
  1.6G

Linux kernel size (2)

Size of Linux source directories (KB)



Linux 2.6.17
Measured with:
du -s --apparent-size



Linux 2.6.29-r46
Measured with:
du -s --apparent-size

Getting Linux sources

- Full tarballs
  - Contain the complete kernel sources
  - Long to download and uncompress, but must be done at least once
  - Example: http://kernel.org/pub/linux/kernel/v2.6/linux-2.6.14.7.tar.bz2

Getting Linux sources

- Incremental patches between versions
  - It assumes you already have a base version and you apply the correct patches in the right order
  - Quick to download and apply
  - Examples
    http://kernel.org/pub/linux/kernel/v2.6/patch-2.6.14.bz2 (2.6.13 to 2.6.14)
    http://kernel.org/pub/linux/kernel/v2.6/patch-2.6.14.7.bz2 (2.6.14 to 2.6.14.7)
- All previous kernel versions are available in http://kernel.org/pub/linux/kernel/

Getting Linux sources

- git
  - `sudo apt-get install git-core6`
  - `git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git linux-2.6`

Getting Linux sources

- bitbake
  - `cd ${OETREE}/build`
  - `bitbake -c clean linux-omap-2.6.28`
  - `bitbake -f -c compile linux-omap-2.6.28`

Top-Level Source Directory

```
arch/
block/
crypto/
Documentation/
drivers/
firmware/
fs/
include/
init/
ipc/
kernel/
lib/
mm/
net/
patches/
samples/
scripts/
security/
sound/
usr/
virt/
```

Embedded Linux usage

Compiling and booting Linux
Kernel configuration

Kernel configuration

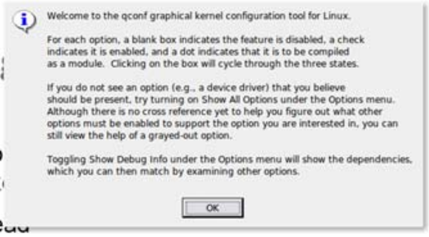
Defines what features to include in the kernel:

- 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-2.6.17-11-generic

make xconfi

`make xconfig`

- The most commo
  to configure the k
- Make sure you rea
  help -> introduction: useful options!
- File browser: easier to load configuration files
- New search interface to look for parameters
- Required Debian / Ubuntu packages:
  libqt3-mt-dev, g++



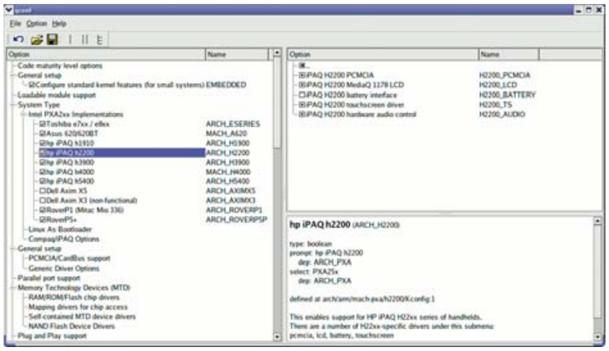
Welcome to the qconf graphical kernel configuration tool for Linux.

For each option, a blank box indicates the feature is disabled, a check indicates it is enabled, and a dot indicates that it is to be compiled as a module. Clicking on the box will cycle through the three states.

If you do not see an option (e.g., a device driver) that you believe should be present, try turning on Show All Options under the Options menu. Although there is no cross reference yet to help you figure out what other options must be enabled to support the option you are interested in, you can still view the help of a grayed-out option.

Toggling Show Debug Info under the Options menu will show the dependencies, which you can then match by examining other options.

make xconfig screenshot



```
Option                                                Name
-Code maturity level options
-General setup
  └☑Configure standard kernel features (for small systems) EMBEDDED
-Loadable module support
-System Type
  -Intel PXA2xx Implementations
    -☑Toshiba e7xx / e8xx                             ARCH_ESERIES
    -☑Asus 620/620BT                                  MACH_A620
    -☑hp iPAQ h1910                                   ARCH_H1900
    -☑hp iPAQ h2200                                   ARCH_H2200
    -☑hp iPAQ h3900                                   ARCH_H3900
    -☑hp iPAQ h4000                                   MACH_H4000
    -☑hp iPAQ h5400                                   ARCH_H5400
    -☐Dell Axim X5                                    ARCH_AXIMX5
    -☐Dell Axim X3 (non-functional)                   ARCH_AXIMX3
    -☑RoverP1 (Mitac Mio 336)                         ARCH_ROVERP1
    -☑RoverP5+                                        ARCH_ROVERP5P
  -Linux As Bootloader
  -Compaq/iPAQ Options
-General setup
  -PCMCIA/CardBus support
  -Generic Driver Options
-Parallel port support
-Memory Technology Devices (MTD)
  -RAM/ROM/Flash chip drivers
  -Mapping drivers for chip access
  -Self-contained MTD device drivers
  -NAND Flash Device Drivers
-Plug and Play support
```

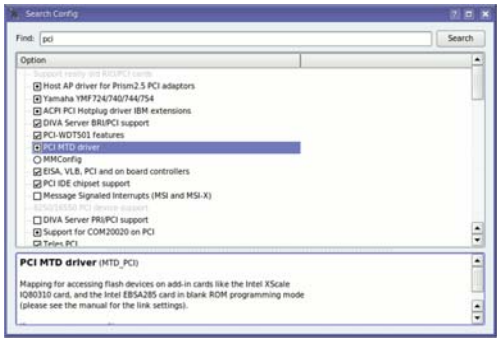
```
Option                                   Name
-☒...
-☒iPAQ H2200 PCMCIA                      H2200_PCMCIA
-☒iPAQ H2200 MediaQ 1178 LCD             H2200_LCD
-☐iPAQ H2200 battery interface           H2200_BATTERY
-☒iPAQ H2200 touchscreen driver          H2200_TS
-☒iPAQ H2200 hardware audio control      H2200_AUDIO
```

hp iPAQ h2200 (ARCH_H2200)

```
type: boolean
prompt: hp iPAQ h2200
  dep: ARCH_PXA
select: PXA25x
  dep: ARCH_PXA

defined at arch/arm/mach-pxa/h2200/Kconfig:1
```

This enables support for HP iPAQ H22xx series of handhelds. There are a number of H22xx-specific drivers under this submenu: pcmcia, lcd, battery, touchscreen

make xconfig search interface



Find: pci

```
Option
-Support really old RIO/PCI cards
  ☒Host AP driver for Prism2.5 PCI adaptors
  ☒Yamaha YMF724/740/744/754
  ☒ACPI PCI Hotplug driver IBM extensions
  ☑DIVA Server BRI/PCI support
  ☑PCI-WDT501 features
  ☒PCI MTD driver
  ○MMConfig
  ☑EISA, VLB, PCI and on board controllers
  ☑PCI IDE chipset support
  ☐Message Signaled Interrupts (MSI and MSI-X)
-8250/16550 PCI device support
  ☐DIVA Server PRI/PCI support
  ☒Support for COM20020 on PCI
  ☑Teles PCI
```

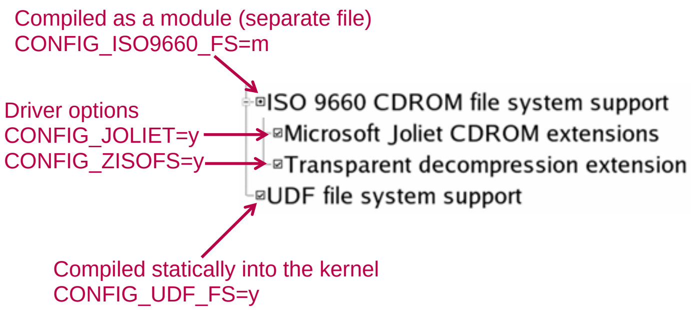
PCI MTD driver (MTD_PCI)

Mapping for accessing flash devices on add-in cards like the Intel XScale IQ80310 card, and the Intel EBSA285 card in blank ROM programming mode (please see the manual for the link settings).

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

```
☒ISO 9660 CDROM file system support
  ☑Microsoft Joliet CDROM extensions
  ☑Transparent decompression extension
☑UDF file system support
```

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_

Kernel option dependencies

- There are dependencies between kernel options
- For example, enabling a network driver requires the network stack to be enabled
- Two types of dependencies
  - *depends on* dependencies. In this case, option A that depends on option B is not visible until option B is enabled
  - *select* dependencies. In this case, with option A depending on option B, when option A is enabled, option B is automatically enabled
  - *make xconfig* allows to see all options, even those that cannot be selected because of missing dependencies. In this case, they are displayed in gray

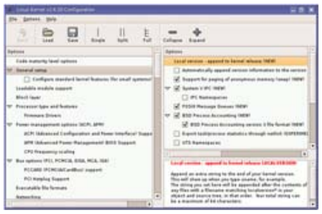
make gconfig

make gconfig

New GTK based graphical configuration interface. Functionality similar to that of make xconfig.

Just lacking a search functionality.

Required Debian packages: libglade2-dev



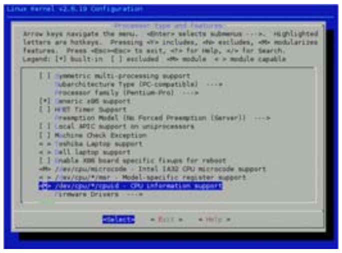
make menuconfig

make menuconfig

Useful when no graphics are available. Pretty convenient too!

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

Required Debian packages: libncurses-dev



```
Linux Kernel v2.6.19 Configuration
                     Processor type and features
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
 letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
 features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
 Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

    [ ] Symmetric multi-processing support
        Subarchitecture Type (PC-compatible)  --->
        Processor family (Pentium-Pro)  --->
    [*] Generic x86 support
    [ ] HPET Timer Support
        Preemption Model (No Forced Preemption (Server))  --->
    [ ] Local APIC support on uniprocessors
    [ ] Machine Check Exception
    < > Toshiba Laptop support
    < > Dell laptop support
    [ ] Enable X86 board specific fixups for reboot
    <M> /dev/cpu/microcode - Intel IA32 CPU microcode support
    < > /dev/cpu/*/msr - Model-specific register support
    <M> /dev/cpu/*/cpuid - CPU information support
        Firmware Drivers  --->

              <Select>    < Exit >    < Help >
```

make oldconfig

make oldconfig

- Needed very often!
- Useful to upgrade a .config file from an earlier kernel release
- Issues warnings for configuration parameters that no longer exist in the new kernel.
- Asks for values for new parameters

If you edit a .config file by hand, it's strongly recommended to run make oldconfig afterwards!

make allnoconfig

make allnoconfig

- Only sets strongly recommended settings to y.
- Sets all other settings to n.
- Very useful in embedded systems to select only the minimum required set of features and drivers.
- Much more convenient than unselecting hundreds of features one by one!

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.

make help

make help

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

Embedded Linux usage

Compiling and installing the kernel
for the host system

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

Slide 69

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

Don’t Use

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.

Don’t Use

All the files in this directory are text files.
Don't hesitate to have a look by yourself!

The Details

To understand a system one must first understand it parts.

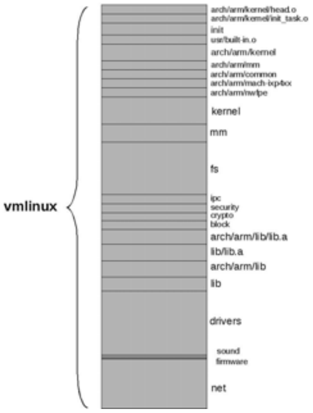
--Chris Hallinan

Link Stage: vmlinux

```
$ arm-linux-ld -EB -p --no-
  undefined -X -o vmlinux
-T arch/arm/kernel/vmlinux.lds \
arch/arm/kernel/head.o        \
arch/arm/kernel/init_task.o   \
init/built-in.o               \
--start-group                 \
usr/built-in.o                \
arch/arm/kernel/built-in.o    \
arch/arm/mm/built-in.o        \
arch/arm/common/built-in.o    \
arch/arm/mach-ixp4xx/built-in.o \
arch/arm/nwfpe/built-in.o     \
kernel/built-in.o             \
mm/built-in.o                 \
fs/built-in.o                 \
ipc/built-in.o           \
security/built-in.o      \
crypto/built-in.o        \
block/built-in.o         \
arch/arm/lib/lib.a       \
lib/lib.a                \
arch/arm/lib/built-in.o \
lib/built-in.o           \
drivers/built-in.o       \
sound/built-in.o         \
firmware/built-in.o      \
net/built-in.o           \
-end-group               \
.tmp_kallsyms2.o         \
```

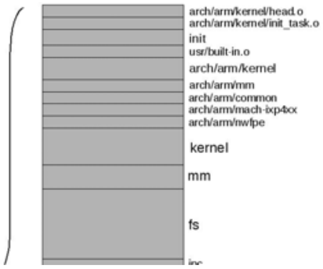
Look in `/usr/local/xtools/arm-unknown-linux-uclibcgnueabi/bin`

vmlinux image components



Compare the two

```
$ arm-linux-ld -EB -p --no-
  undefined -X -o vmlinux
-T arch/arm/kernel/vmlinux.lds \
arch/arm/kernel/head.o        \
arch/arm/kernel/init_task.o   \
init/built-in.o               \
--start-group                 \
usr/built-in.o                \
arch/arm/kernel/built-in.o    \
arch/arm/mm/built-in.o        \
arch/arm/common/built-in.o    \
arch/arm/mach-ixp4xx/built-in.o \
arch/arm/nwfpe/built-in.o     \
kernel/built-in.o             \
mm/built-in.o                 \
fs/built-in.o                 \
```



vmlinux Image Components Description

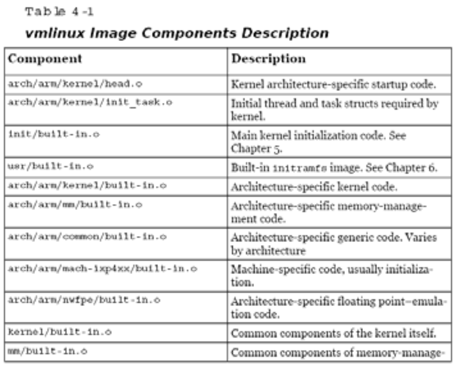


Table 4-1

vmlinux Image Components Description

Component	Description
arch/arm/kernel/head.o	Kernel architecture-specific startup code.
arch/arm/kernel/init_task.o	Initial thread and task structs required by kernel.
init/built-in.o	Main kernel initialization code. See Chapter 5.
usr/built-in.o	Built-in initramfs image. See Chapter 6.
arch/arm/kernel/built-in.o	Architecture-specific kernel code.
arch/arm/mm/built-in.o	Architecture-specific memory-management code.
arch/arm/common/built-in.o	Architecture-specific generic code. Varies by architecture
arch/arm/mach-ixp4xx/built-in.o	Machine-specific code, usually initialization.
arch/arm/nwfpe/built-in.o	Architecture-specific floating point–emulation code.
kernel/built-in.o	Common components of the kernel itself.
mm/built-in.o	Common components of memory-manage-