New features in Linux 2.6

New features in Linux 2.6

Since 2.6.10

Thomas Petazzoni / Michael Opdenacker

Free Electrons

http://free-electrons.com/

Created with OpenOffice.org 2.x

Rights to copy

Document sources, updates and translations:
http://free-electrons.com/docs/linux26-features

Corrections, suggestions, contributions and translations are welcome!

Best viewed with...

This document is best viewed with a recent PDF reader or with OpenOffice.org itself!

- Take advantage of internal or external hyperlinks. So, don't hesitate to click on them!
- Find pages quickly thanks to automatic search
- Use thumbnails to navigate in the document in a quick way

If you're reading a paper or HTML copy, you should get your copy in PDF or OpenOffice.org format on http://free-electrons.com/training/devtools!

New features in 2.6

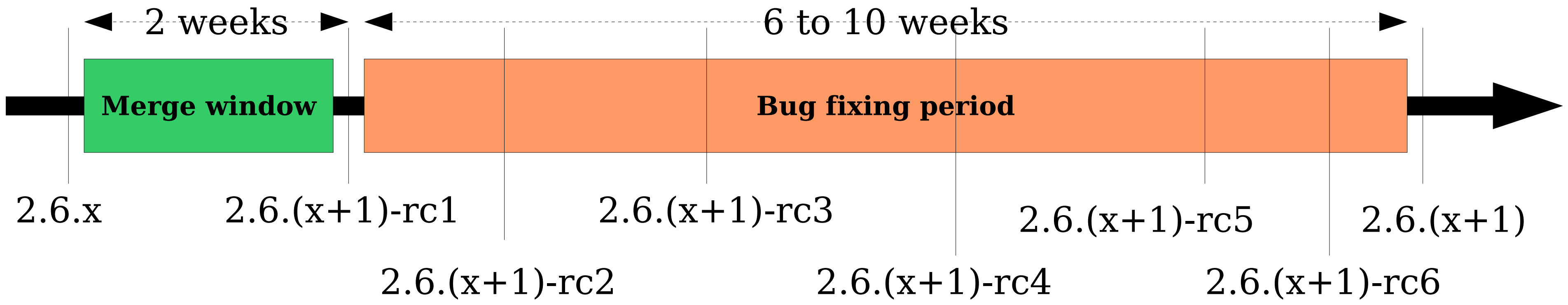
- This presentation covers the new features that have been added to the Linux kernel since 2.6.10
- Only a selection of the major features is covered
- Emphasis on embedded-related features
- Based on the very valuable LinuxChanges initiative of the KernelNewbies project
  http://kernelnewbies.org/LinuxChanges

Development model

- The previous development/stable versions development model has been dropped
- The 2.6 branch exists since 2003, and no 2.7 development branch has been opened
- New features are gradually added in each version of the kernel
  - Fits better the distributors needs (no need to backport a lot of features from development kernels)
  - Nicer for kernel developers : their changes are sooner made available to users

Development model (2)

- After the release of a 2.6.x kernel
  - Two weeks of *merge window* to integrate new features, drivers, etc. Closed by the release of 2.6.(x+1)-rc1
  - Six to ten weeks bug-fixing period, with regular releases of 2.6.(x+1)-rcY kernels
- Release of 2.6.(x+1) once considered stable



Source code management tool (1)

- Until 2002 : no central source code management tool
- In 2002, switch to BitKeeper
  - During the 2.5 development cycle
  - BitKeeper was a distributed SCM, that suited well the development model of the kernel. Free version available, but proprietary.
- In April 2005, announcement that BitKeeper will no longer be free
  - Torvalds starts the development of a new distributed SCM, Git
  - Git is now used by the kernel and many other free software projects
  - Maintained by Junio Hamano.

Source code management tool (2)

- Git is now used by most core kernel developers
- Allows to get an higher number of patches integrated
  - Kernel development is going on with a very fast rhythm
  - 5000 to 7000 patches in each kernel version
  - 85 lines added to the kernel every hour, since 3 years
- Distributed nature well-suited for free software projects
  - Local branches for development
  - Exchange development versions between developers
  - Offline development
  - Powerful merging facilities

Generic IRQ subsystem

- A new, generic IRQ subsystem
- Fit the needs of all architectures, including ARM
- Cleaner conception, by distinguishing
  - Flow type management
    (level-triggered, edge-triggered, simple and per-cpu interrupts)
  - Low-level chip-dependent management (`irq_chip` structure)
- Reduce code duplication
- Ease the addition of new boards and architectures
- Preliminary work by Ingo Molnar in 2.6.10,
  completed in 2.6.18 by Thomas Gleixner.

Preemption

- 2.6.11, Big Kernel Lock preemption
  - Old mutual exclusion mechanism introduced before SMP was common
  - Causes latency issues, so being replaced by fine-grained locks
  - BLK preemption turns the BLK into a semaphore instead of a spinlock
- 2.6.13, Voluntary preemption
  - Add explicit preemption points in the kernel,
    using the `might_sleep()` function
  - Latency reduction with fewer throughput reduction
  - `PREEMPT_VOLUNTARY` in `kernel/Kconfig.preempt`

I/O scheduler work

- In 2.6.10
  - I/O schedulers can be compiled as modules
  - I/O schedulers can be changed on the fly
- Improve fairness and correctness of the CFQ I/O scheduler
- Introduction of I/O priorities in CFQ
  - Two new system calls: `ioprio_set()`, `ioprio_get()`
  - Userspace program: `ionice`
- In 2.6.18, CFQ becomes the default I/O scheduler

Kobject events userspace notification

- Allow userspace to be notified of kernel events occurring on kobjectfs
- Uses a netlink socket
- Userspace can be notified of events such as
  - Device insertion and removal
  - Mount notifications
  - Simple events (CPU overheating, etc.)
- Initial version based on `/sbin/hotplug`
- Foundation for udev

Four level page tables

- Linux generic memory management code relied on three level page tables
  - PGD -> PMD -> PTE
  - The architecture-dependent code is responsible for making this three level scheme to match with what the hardware provides
- The new x86_64 supports four level pages tables, which cannot be supported by the generic code
- Introduction of a new level, between PGD and PMD, called PUD
  - PGD -> PUD -> PMD -> PTE
- Virtual address space grown from 512 GB to 128 TB

New architectures

- FRV
  - Futjitsu 32 bits processors, VLIW architecture
  - Support for MMU and non-MMU versions
- AVR32
  - Atmel 32 bits processors
- Blackfin
  - 32 bits microprocessor/DSP, no MMU
  - Support coming from the uClibc project
- Xtensa
  - 32 bits processor by Tensilica

debugfs

- Virtual filesystem making it easy to export kernel information to userspace
- Preferred way compared to `/proc` entries or `/sys` entries
- Very simple API

```
static char *acme_buf;
static unsigned long acme_bufsize;
static struct debugfs_blob_wrapper acme_blob;
static struct dentry *acme_buf_dentry;

/* Module init */
acme_blob.data = acme_buf;
acme_blob.size = acme_bufsize;
acme_buf_dentry = debugfs_create_blob("acme_buf", S_IRUGO,
                                          NULL, &acme_blob);

/* Module exit */
debugfs_remove (acme_buf_dentry);
```

Infiniband

- Switched fabric communication link
- Used in high performance computing
- High-throughput, from 2 to 100 Gbit/s using aggregation
- Low latency, around 200 nanoseconds
- In the kernel
  - Hardware drivers
  - SRP, SCSI RDMA Protocol
  - IPoIB, IP over Infiniband
  - kDAPL, RDMA API
  - Interface to userspace

CPU sets

- Big NUMA systems present challenges for the efficient scheduling and memory placement of processes
- On small systems, CPU affinity might be sufficient, but not on big NUMA systems
- Allows to create cpusets, consisting of
  - CPU nodes
  - Memory nodes
- And then to assign tasks to a given cpuset
- User interface in the form of a virtual cpuset filesystem

eXecute In Place

- Add support for execution of userspace programs directly from Flash
- Instead of loading them in RAM inside the page cache, use direct references to their location in ROM
- Only works devices that are CPU-addressable at all times
- The block device driver must support the `direct_access()` operation
- The filesystem must support XIP (only ext2 supported)
- Use the `-o xip` option to mount
- Doesn't seem to be widely used, designed for S390, not embedded devices

Tick frequency

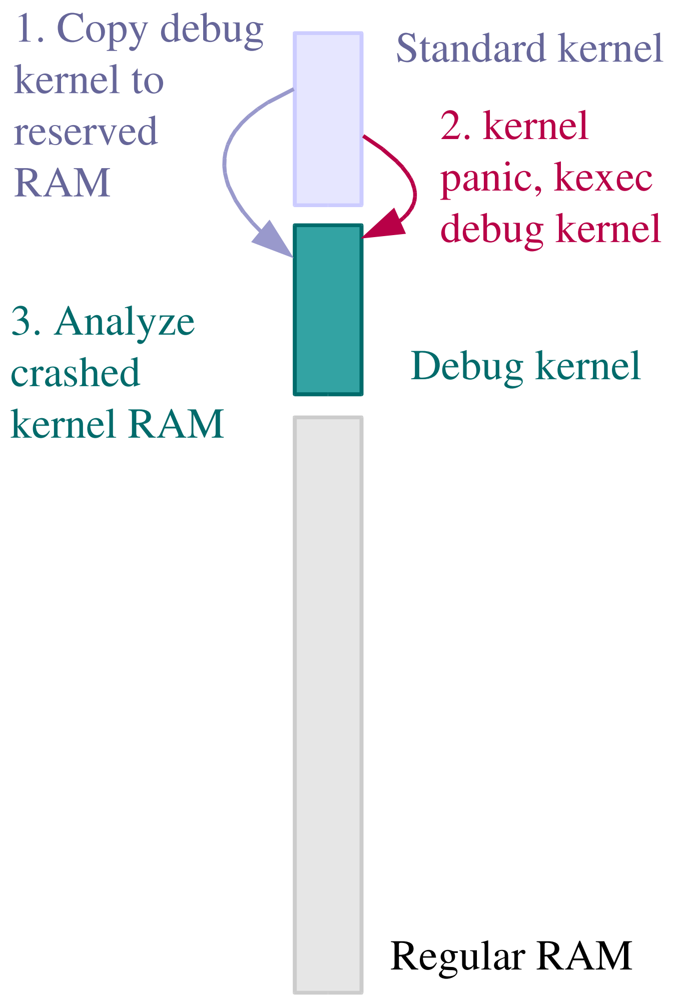
- In 2.4, the tick frequency was 100 Hz
- In 2.6, it was increased to 1000 Hz
- In 2.6.13, it was reduced to 250 Hz, but made configurable
- In 2.6.21, the Dynamic Ticks patch is merged
  - It removes the regular timer tick
  - Relies on programming the hardware timer so that the system is only woken up for the next event
  - Allows to reduce power consumption
  - Initial support only for x86
  - Extended to x86_64, PPC, ARM and MIPS in 2.6.24

inotify

- Provide filesystem events notification to userspace programs
- Replacement for `dnotify`
  - Relied on one file descriptor for each monitored directory. Issues with the maximum number of file descriptors, and removable devices, awful userspace interface
  - Can only monitor changes at the directory level, not file level
- Notification of events such as access, modification, change of attributes, open, close, move, rename, creation, deletion
- Mostly used for desktop search system so that reindexing is not needed
- Might be useful for other applications as well

Kernel crash analysis with kexec

- `kexec` system call: makes it possible to call a new kernel, without rebooting and going through the BIOS / firmware.
- Idea: after a kernel panic, make the kernel automatically execute a new, clean kernel from a reserved location in RAM, to perform post-mortem analysis of the memory of the crashed kernel.
- See Documentation/kdump/kdump.txt in the kernel sources for details.


1. Copy debug kernel to reserved RAM
Standard kernel
2. kernel panic, kexec debug kernel
3. Analyze crashed kernel RAM
Debug kernel
Regular RAM

devfs removal

- Devfs was a virtual filesystem added in the 2.5 development branch
- Its goal was to provide a dynamic `/dev` directory, instead of having thousands of static and useless entries for every possible device
- For several reasons, it was disliked by many kernel developers
  - Broken naming
  - Naming policy in the kernel
  - Not flexible enough
- Finally removed in 2.6.18, superseded by udev.

Networking

- DCCP
  - Datagram Congestion Protocol, layer 4
  - Protocol that implements bidirectional, unicast connections of congestion-controlled, unreliable datagrams
- TCP congestion work
  - New TCP congestion algorithms, suited for very high speed networks or low quality networks
- SCTP
  - Stream Control Transmission Protocol, layer 4
  - Multi-streaming, out-of-order reliable, protocol

Soft lockup detection

- `CONFIG_DETECT_SOFTLOCKUP`
- Per-cpu watchdog threads are started, and if not scheduled during more than 10 seconds, a debug message is printed
- Allows to debug deadlocks generated by incorrect locking based on spinlocks

Netlink connector

- Simplified API to use the netlink socket facility from the kernel point of view
- Receive messages through a callback
  ```
  int cn_add_callback(struct cb_id *id, char
  *name, void (*callback) (void *));
  ```
- Send messages
  ```
  void cn_netlink_send(struct cn_msg *msg, u32
  __groups, int gfp_mask);
  ```
- Useful for bi-directional exchange of information between a userspace application and the kernel
- See Documentation/connector/connector.txt

FUSE

- Filesystem in userspace
- Allows to write filesystem drivers in userspace, using `libfuse`
- These filesystems can be used like traditional filesystem: mounted and accessed using the standard API
- The fuse kernel module *forwards* VFS calls to an userspace program
- Wide variety of userspace filesystem drivers: sshfs, ntfs-3g, zfs, gmailfs, etc.

Relay

- Mechanism that allows kernel code to send large amount of informations to userspace in a fast and efficient way
- Per-cpu buffers inside the kernel are visible from userspace as files that be read, or better, mapped, with the regular API
- Unidirectional only, kernel to user
- Used by the *blktrace* logging facility
- See Documentation/filesystems/relay.txt

Shared subtrees

- New mounting semantics
  - Shared mount, can be replicated to many mountpoints and replicas remains the same
  - Slave mount, same as shared mount, except that mount and unmount events only propagate towards it
  - Private mount, the regular mounting semantic
  - Unbindable mount, cannot be binded
- `mount --make-shared, --make-slave, --make-private, --make-unbindable`, and their recursive counterparts
- See Documentation/sharedsubtree.txt

Shared subtrees: use cases

- A process wants to clone its own namespace, but still wants to access the CD that got mounted recently
  - `mount --make-shared /cdrom`
- A process wants its mounts invisible to any other process, but still be able to see the other system mounts
  - `mount --make-rshared /`
  - `mount –make-rslave /myprivatetree`
- Per-user namespaces
- Versionned file-systems

Process Events Connector

- Mechanism to notify userspace applications of events concerning processes occurring inside the kernel : fork, exec, id change, exit
- Relies on the netlink connector facility
- Use cases
  - Auditing
  - Activity monitoring
  - Security

ARM

- CPU hotplug
- New CPUs
  - ARM9TDMI, ARM7TDMI, ARM740T, ARM940T, Micrel/Kendin KS8695, ARM V7, Atmel AT91SAMxxx, Samsung S3C2443, op13xx, ARM946E-S, etc.
- New boards
  - Too many to name them all
- New drivers
  - Too many to name them all

New system calls (1)

- `*at()` system calls
  - `openat(), mkdirat(), mknodat(), fchownat(), futimesat(), fstatat(), unlinkat(), renameat(), linkat(), symlink_at(), readlinkat(), fchmodat(), faccessat()`
  - Operations relative to an open file descriptor corresponding to a directory
  - Needed to implement race-free filesystem traversal
  - Needed to implement virtual per-thread current directory, for backup software.

New system calls (2)

- `pselect(), ppoll()`
  - Signal-aware version of `select()` and `poll()`
  - No more race condition or complicated self-pipe trick needed to properly handle signals with `select()` and `poll()`
  - See http://lwn.net/Articles/176911/
- `unshare()`
  - Allows to selectively unshare resources that are normally shared between threads of the same process
  - Polyinstantiated directories, for security
  - Intra-application isolation
  - See Documentation/unshare.txt

New system calls (3)

- `tee()`, `splice()`, `vmsplice()`
  - `splice()` allows to moves data from one file descriptor to another file descriptor without involving userspace
  - `tee()` does the same, except that it doesn't consume the input data, it can still be read from userspace
  - `vmsplice()` allows to feed userspace memory into a pipe
  - Increase performances by decreasing the number of memory copies
  - See `tee(2)`, `splice(2)` and `vmsplice(2)`
- `sync_file_range()`
  - Allows to synchronize part of a file to the disk, and wait for the completion of part of the process, depending on given flags.

New system calls (4)

- `fallocate()`
  - Allows to preallocate or deallocate blocks on disk
  - Ensure that blocks are contiguous on disk, which can improve performances
  - Only implemented in ext4 and OCFS2 so far

Mutex primitive

- Kernel only provided a semaphore locking facility
- Most of them were used as binary semaphores for mutual exclusion, and a cheaper implementation is possible for binary semaphores (smaller structure and code, faster)
- Introduction of the mutex API and progressive conversion of the code base
- `mutex_init()`, `mutex_lock()` and variants, `mutex_unlock()`, `mutex_is_locked()`
- See Documentation/mutex-design.txt

High resolution timers

- Classic timers in Linux have at most a resolution equal to the timer frequency (1 ms, 4 ms or 10 ms)
  - Resolution not sufficient for multimedia or realtime applications
- High resolution timers allows to have a much higher resolution, depending on what the hardware allows
- Provide a new in-kernel API, also used to improve the userspace time-related functions
  - Nanosleep
  - Itimers
  - POSIX timers

SLOB / SLUB allocators

- Historically, the Linux kernel relies on an allocator using the SLAB strategy for small allocations
- In 2.6, two new alternative allocators, also implementing the SLAB strategy have been added
  - SLOB, with a focus on code size and low memory overhead, for embedded systems
  - SLUB, with a focus on scalability while remaining general-purpose, now the default allocator

Size reduction

- `CONFIG_EMBEDDED`
  - Provides several suboptions to disable support for various kernel functionalities that may not be useful on embedded systems
- `CONFIG_BLOCK`
  - Allows to completely disable the block layer of the kernel, useless when using the MTD layer
  - Useful for embedded systems
- Linux-Tiny
  - Set of patches started by Matt Mackall to reduce kernel size
  - Michael Opdenacker is the new maintainer, cleanup and mainline merge effort taking place.

configfs

- New virtual filesystem
- Aimed at easing the configuration of complex kernel applications from userspace
- When directories and files are created by userspace applications, callbacks are called in the kernel
- Allows greater flexibility of configuration, without using ioctl() calls
- Used by DLM, OCFS and netconsole for the moment
- See http://lwn.net/Articles/148973/

Clustering

- OCFS 2
  - Clustering filesystem developed by Oracle. Allows several hosts to access the same storage by taking care of concurrent accesses
- GFS
  - Clustering filesystem originally developed by Sistina, bought by RedHat and released under the GPL
- TIPC
  - Transparent Inter Process Communication protocol
  - Intra cluster communication
  - Originates from Ericsson

blktrace

- Kernel module and userspace tool to trace I/O requests
- All the steps of an I/O request can be traced
- Useful to debug issues with I/O loads
- Relies on relayfs to transfer large amount of informations from the kernel to userspace

```
% blktrace -d /dev/sda -o - | blkparse -i -
  8,0    3        1     0.000000000   697 G W 223490 + 8 [kjournald]
  8,0    3        2     0.000001829   697 P R [kjournald]
  8,0    3        3     0.000002197   697 Q W 223490 + 8 [kjournald]
  8,0    3        4     0.000005533   697 M W 223498 + 8 [kjournald]
  8,0    3        5     0.000008607   697 M W 223506 + 8 [kjournald]
  8,0    3        6     0.000011569   697 M W 223514 + 8 [kjournald]
```

Userspace priority inheritance

- Priority inversion occurs when a low-priority process holds a lock needed by an higher-priority process
- A classical solution is to temporarily boost the low-priority process's priority to the high-priority process's priority, until it releases the lock : priority inheritance
- Support for priority inheritance in userspace locks has been added in 2.6.18, through the *futex* facility
- Can be used using `pthread_mutexattr_setprotocol (..., PTHREAD_PRIO_INHERIT)`

Lockdep

- Kernel lock validator
- Allows to detect deadlock before they occur, even rare deadlocks
- Associate each spinlock with a key, so that similar locks are handled only once
- When locking, look at all already taken locks, and make sure that none of these locks are taken after the newly taken lock in other contexts.
- When unlocking, make sure that the lock being unlocked is at the top of the taken locks.
- Validate spinlocks vs interrupts behavior.

ext4 (1)

- New major developments had to be made on the ext3 filesystem
  - Did not match the stability requirements of ext3
  - A new ext4 filesystem has been created, initially a simple copy of ext3
- Goal is to improve the scalability and reliability of the ext filesystem
  - Large filesystems (64 bits)
  - Challenge of the growing size of hard disks but not their throughput or access time

ext4 (2)

- Features
  - Extents
  - Filesystems bigger than 16 TB
  - Internal redundancy to improve behavior in the face of corruption
  - Improved file allocation
  - Large inodes (nanoseconds timestamps, etc.)
  - Reduced fsck time
- Not production-ready
- See Documentation/filesystems/ext4.txt

eCryptfs

- « POSIX-compliant enterprise-class stacked cryptographic filesystem »
- Stacked on top of existing filesystems to provide on the fly encryption and decryption
- Other cryptographic approach at the block level
  - dm-crypt, Loop-AES, TruCrypt

Libata PATA

- New SATA devices and controllers are very similar to SCSI devices and controllers
  - From the origin, SATA support has been integrated inside the SCSI stack, using libata
- libata PATA is an effort to propagate this idea to traditional PATA devices and controllers
- In the end, all devices should appear as SCSI devices, with a single API
- Production-ready, already used on modern distributions
- See Documentation/DocBook/libata

OSS removal

- Open Sound System
  - Original sound subsystem of the Linux kernel
  - Now superseded by ALSA
- OSS drivers are slowly removed, when an ALSA driver exists for the same driver
- Should be completely dropped in a couple of years
- New drivers should be developed inside the ALSA framework

Namespaces

- One method of virtualization is isolation
  - Single kernel, but processes are enclosed in isolated environments
  - Approach used by vserver and OpenVZ
- Needs support from all the kernel subsystems
  - User namespaces, 2.6.23
  - PID and network namespaces, 2.6.24
  - New `clone()` flags
  - More subsystem namespaces in next versions of the kernel
- Also useful for checkpoint and restart of processes in HPC

Vectored AIO support

- The Linux kernel supports asynchronous I/O operations
- Until 2.6.19, an AIO operation could only work on a single buffer
- Since 2.6.19, a read of write operation can be made at once on several userspace buffers

Paravirtualization

- Paravirtualization
  - Running a guest kernel under a host kernel, where the guest has been modified to run on a virtual architecture, similar to the hardware architecture
  - The guest kernel must call an hypervisor for privileged operations (enable/disabling interrupts, for example)
- Generic paravirtualization support, introduced in 2.6.20
  - Encapsulate the set of operations for which a call to an hypervisor has to be made into a set of functions pointers, *paravirt_ops*
  - Defaults to their normal behavior, but can be modified by the hypervisor
- Generic paravirtualization support, 2.6.20
- Xen, 2.6.23
- Lguest, 2.6.23

Paravirtualization (2)

- Xen, 2.6.23
  - Parts of Xen have been merged
  - Only guest support, no hypervisor, no dom0
  - Based on `paravirt_ops`
- Lguest, 2.6.23
  - Simple hypervisor for Linux on Linux
  - Demonstrate the how powerful the `paravirt_ops` infrastructure is
  - Hypervisor in a kernel module
  - Specifically-compiled guest kernels
  - See Documentation/lguest

Paravirtualization (3)

- Virtual Machine Interface, 2.6.21
  - Proposed by VMware
  - The `paravirt_ops` approach was preferred
  - The port of VMI on top of `paravirt_ops` has been merged in 2.6.21
  - Another interface between guest kernels and an hypervisor
  - Allows Linux guest kernels to run on VMware products

KVM

- Driver for Intel and AMD hardware virtualization extensions
- Export them to userspace through `/dev/kvm`
- Using this driver, an userspace process can run a guest virtual machine with fully virtualized hardware
- Virtualizes CPU and memory,
  but the rest of the hardware has to be emulated
- Works in combination with Qemu
- Runs unmodified guests
- Also supports a paravirtualized mode, migration, etc.
- See http://kvm.qumranet.com/

Fault injection

- Framework for injecting faults in a running kernel
- The goal is to test error code paths
- Allows to
  - Make SLAB allocations fail
  - Make page allocations fail
  - Return I/O errors
- Userspace configuration through debugfs
  - Set the rate of faults
  - Filter the injection to certain tasks or particular pieces of code
- See Documentation/fault-injection/fault-injection.txt

Generic HID layer

- Human Interface Devices
- Allows to connect keyboards, mice, joysticks, graphic tablets
- Standard protocol, initially used only for USB devices
- The Linux HID layer has been improved to be more generic
- Support HID devices on any buses, and particularly Bluetooth

ALSA System-on-Chip

- Improved support for sound processors on embedded systems
- Advantages
  - Reuse codec drivers on other platforms
  - Reuse of platform specific audio drivers on different machines
  - Easy I2S/PCM digital audio interface configuration
  - Allow machines to add controls and operations to the audio subsystem
  - Power management
- See http://www.rpsys.net/openzaurus/patches/alsa/info.html

GPIO API

- Simple and minimalist programming interface for General Purpose Input/Output
- Driven by the need of portability of drivers between architectures (ARM and AVR32)
- Having an unified API allows other developers to more easily read the code of others
- See Documentation/gpio.txt

Wireless stack

- Huge differences between wireless adapters in the range of functionalities supported in hardware and those that have to be implemented in software
  - Mess and duplication in Linux drivers
- The Devicescape company developed and released under the GPL a new wireless stack, now integrated to the kernel
- Complete software MAC, WEP, WPA, link-layer bridge module, hostapd, QoS, etc.
- New kernel/user interface, based on netlink instead of ioctl()
- Kernel drivers must be rewritten,
  but will be cleaner and simpler thanks to the new stack.

Firewire stack

- New Firewire stack being developed
- *« get a small, maintainable and supportable FireWire stack »*
- Smaller code base and binaries
- Cleaned-up and improved in-stack APIs and design
- Consolidation of the four userspace ABIs into a single improved ABI
- Solve bugs and security issues
- Compatibility at the library level (libraw1394, libdc1394)
- Both stacks are in the kernel, until the new one fully replaces the old one.

UBI

- « Unsorted Block Images »
- Volume management system for flash devices
  - Manages multiple volumes on a single flash
  - Spreads the I/O load across the whole flash chip (global wear-leveling)
  - Transparently handles bad physical erase blocks
  - Handles bit-flips in the Flash by remapping blocks when bit-flips are detected
  - Atomic logical eraseblock change
- See http://www.linux-mtd.infradead.org/doc/ubi.html

CFS (1)

- During the 2.5 cycle, an O(1) scheduler, developed by Ingo Molnar, was merged
  - Scalability improvements with many processes
- Issue with O(1) scheduler and interactive tasks, that did not react quickly enough in presence of CPU-bound tasks
  - Interactivity estimator and heuristics developed by Con Kolivas
- Con Kolivas developed a brand new scheduler, called RDSL, based on a strict fairness
- Ingo Molnar took over the idea, and developed CFS, Completely Fair Scheduler, which was finally merged.

CFS (2)

- In the O(1) scheduler, two arrays of run queues to keep track of processes
  - Each array with 140 entries to take priorities into account
- In the CFS, all runnable processes are stored in a red-black tree
- The key in the red-black tree is the time
  that the process should have had on the CPU
  - Basically the time for which the process waited without being run, divided by the number of processes, with a correction for priority
  - Allows to be perfectly fair with processes, and be nice with interactive processes
- Introduction of modular, chained, schedulers

Lumpy reclaim, anti-fragmentation

- Fragmentation of physical memory quickly becomes an issue for big allocations on highly-loaded machines
- Several improvements made in this area
  - Lumpy reclaim
    - Tune the page reclaim algorithm to not only take the recency into account, but also the fact that pages are grouped or not
  - Anti-fragmentation
    - Grouping pages of related type together (movable, reclaimable, unreclaimable, highatomic)
    - When pages are migrated or reclaimed, large regions of physical memory are freed, allowing bigger allocations to succeed

UIO

- Framework that allows the development of device drivers in userspace
- Small kernel module to register the device and for the interrupt handler, the rest in userspace
- Easier to debug
- Driver can be kept proprietary with no licensing issues
- Interface: `/dev/uioX`
  - `mmap()` to read/write registers
  - Blocking `read()` to be notified of interrupts
- http://www.free-electrons.com/kerneldoc/latest/DocBook/uio-howto/

USB authorization

- Allows to control whether an USB device can be used or not in the system
- Needed for Wireless USB
- User interface through `/sys`
  - Default behavior for an USB controller (authorize or not new devices to be used)
  - Selectively authorize devices

Task control groups

- Mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behavior
- Used by
  - The new CFS scheduler for group scheduling
  - Cpusets to associate CPU and memory nodes to a set of tasks
- Uses a new virtual filesystem as the userspace/kernel API
- See Documentation/cgroups.txt

Linux Kernel Markers

- Static probing points inserted at various points inside the kernel
- Well-defined trace points at correct places in the kernel
  - Added by subsystem developers
  - Maintained in the official kernel tree
- Allow users and system administrators to use higher-level tools
  - SystemTap
  - LTTng
- See http://lwn.net/Articles/245671/ and Documentation/markers.txt

Power management

- Cpuidle
  - Generic framework for supporting software-controlled idle processor power management
  - Hardware specific drivers
  - Various governing for the state transition decisions
- PowerTop
  - Userspace tool showing the current consumption and the processes that are waking up the processor
- Latency and power management
  - Framework for expressing latency constraints, and make sure that they are taken into account for power management decisions

LZO support for JFFS2

- JFFS2 compression is classically based on zlib
- In 2.6.24, support was added for LZO compression, which generally works better than zlib

Soon in a kernel near you...

Filesystems (1)

- BTRFS
  - New generation filesystem developed by Oracle to compete with Sun's ZFS (checksumming, snapshot, volumes, mirroring, striping, extent-based, etc.)
  - Can be associated with CRFS to export filesystems over the network
  - See http://oss.oracle.com/projects/btrfs/
- LogFS
  - Flash filesystem, aimed at replacing JFFS2
  - Focus on scalability, usable on large devices
  - See http://www.logfs.org/logfs/

Filesystems (2)

- UnionFS
  - Stackable unification filesystem
  - Merge the contents of several directories, while keeping their physical contents separate
  - Useful to merge a small read-write device over a read-only device, for example for LiveCDs or embedded devices
  - See http://www.am-utils.org/project-unionfs.html
- ChunkFS
  - Experimental, chunk-based filesystem,
    designed to work properly on vast amounts of storage
  - See http://www.valhenson.org/chunkfs/

Utrace

- Infrastructure for tracing and controlling user threads
- Foundation for writing tracing engines, which can be loaded as modules
- Provides three facilities
  - Thread event reporting
  - Core thread control
  - Thread machine state access
- Replacement for ptrace(), with a similar userspace API

Real Time improvements

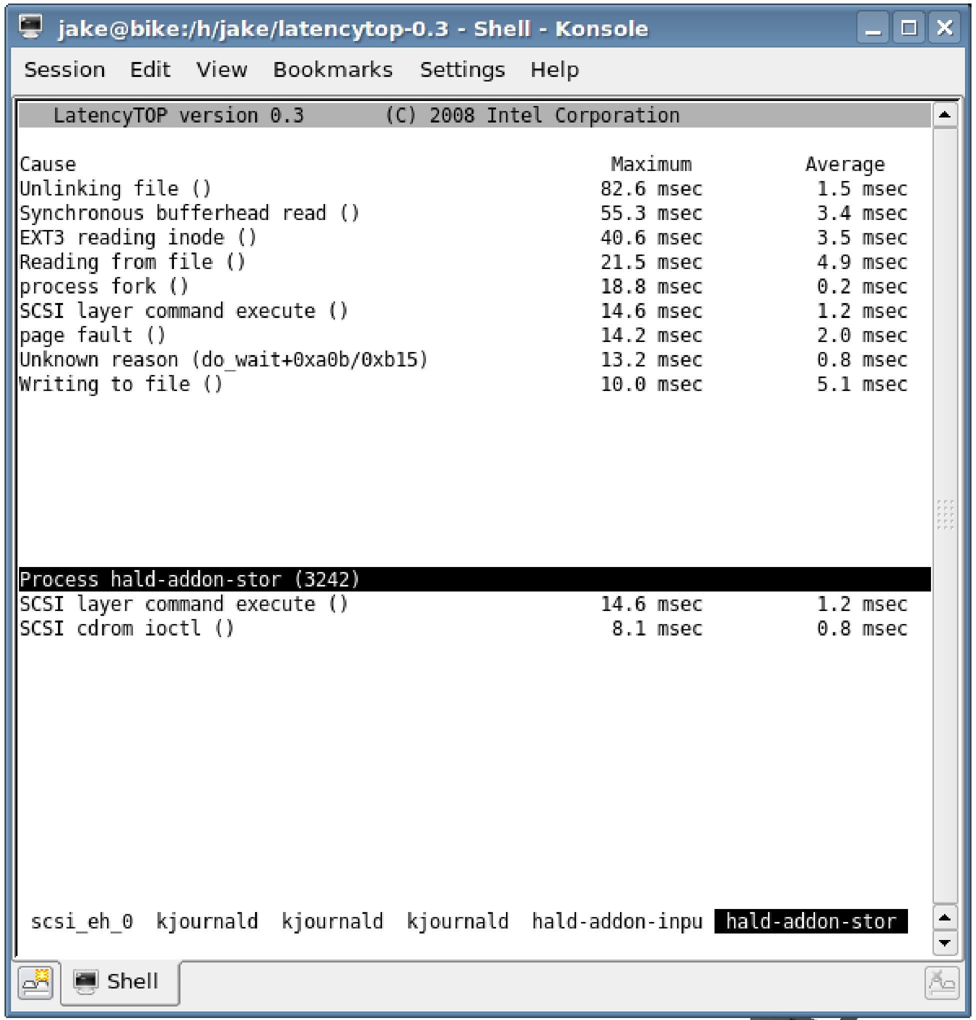
- Real-time mutex code
  - Turn all spinlocks to mutexes to make them preemptible
- Threaded interrupt handlers
  - Interrupt handlers running in threads allows them to be scheduled and prioritized like any other threads
  - Allows low-priority interrupts to not cause latencies on high-priority tasks
- Preemptible Read-Copy-Update mechanism
- Other latency reduction patches
- More than 400 patches still in -rt

CAN protocol

- Support for the Controller Area Network specification
- Contributed by Volkswagen
- Usable through the regular socket API, thanks to a new protocol type, `PF_CAN`

LatencyTop support

- Userspace tool and kernel support to detect where the latencies are
- Accumulate the time spent sleeping between two points, to find where the latencies come from
- See http://www.latencytop.org/


jake@bike:/h/jake/latencytop-0.3 - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

```
  LatencyTOP version 0.3        (C) 2008 Intel Corporation

Cause                                          Maximum          Average
Unlinking file ()                              82.6 msec         1.5 msec
Synchronous bufferhead read ()                 55.3 msec         3.4 msec
EXT3 reading inode ()                          40.6 msec         3.5 msec
Reading from file ()                           21.5 msec         4.9 msec
process fork ()                                18.8 msec         0.2 msec
SCSI layer command execute ()                  14.6 msec         1.2 msec
page fault ()                                  14.2 msec         2.0 msec
Unknown reason (do_wait+0xa0b/0xb15)           13.2 msec         0.8 msec
Writing to file ()                             10.0 msec         5.1 msec

Process hald-addon-stor (3242)
SCSI layer command execute ()                  14.6 msec         1.2 msec
SCSI cdrom ioctl ()                             8.1 msec         0.8 msec

 scsi_eh_0  kjournald  kjournald  kjournald  hald-addon-inpu  hald-addon-stor
```

Syslets

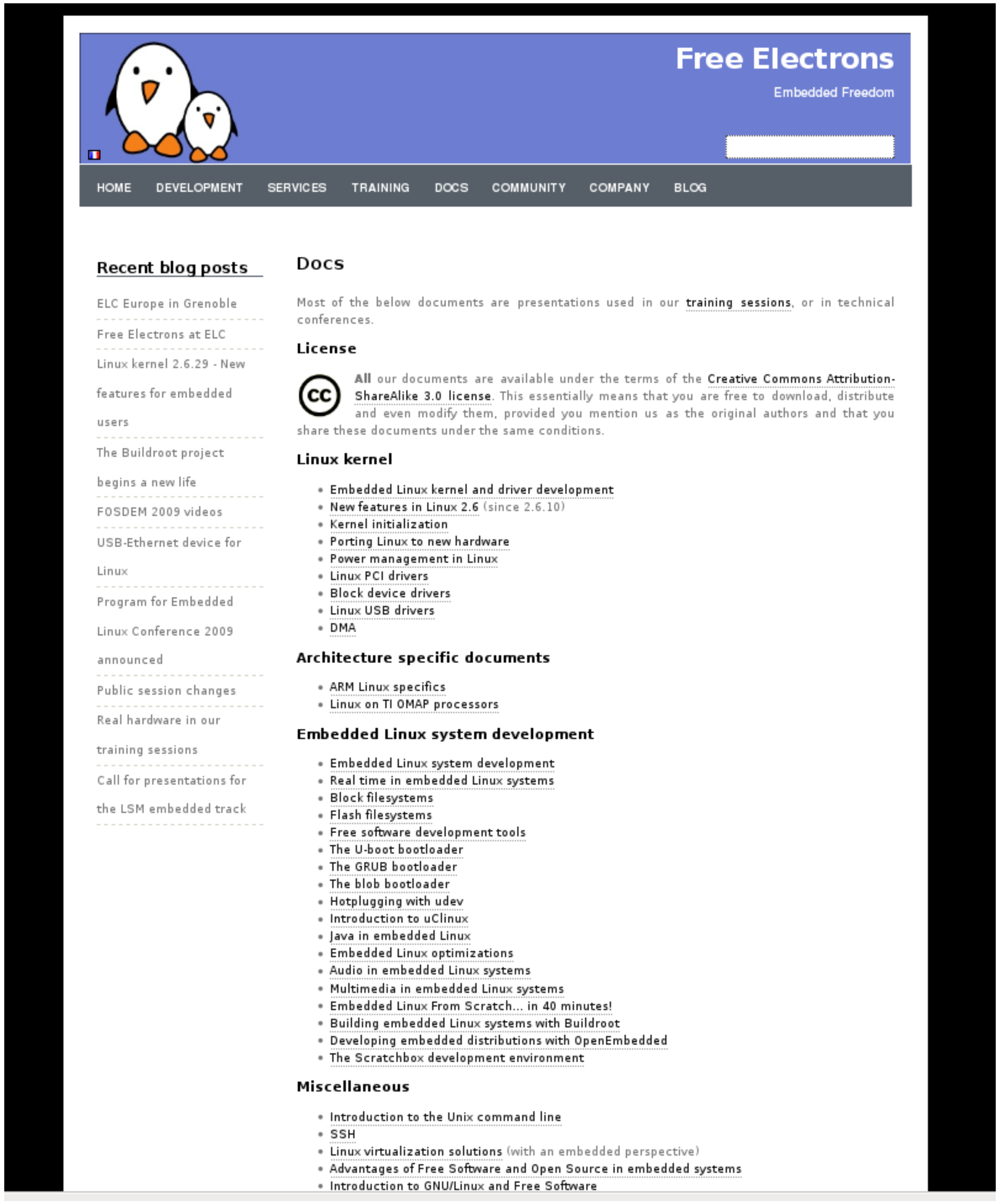
- Syslets are a mechanism to perform asynchronous system calls from userspace
- Instead of developing an ad-hoc asynchronous version of each system call, syslets is a generic solution
- Executes a system call, and if it happens to block, creates a new thread to continue execution in userspace while the system call is being executed in the original thread
- Complex version proposed initially,
  with multiple syscalls in one functionality.
- Simpler version being proposed now, to ease inclusion

KGDB

http://kgdb.linsyssoft.com/

- The execution of the kernel is fully controlled by `gdb` from another machine, connected through a serial line.
- Can do almost everything, including inserting breakpoints in interrupt handlers.
- A simplest version of Linsyssoft patch is being worked on by Ingo Molnar for integration inside the kernel, but Linus Torvalds is known not to like debuggers a lot.

Related documents



Recent blog posts

- ELC Europe in Grenoble
- Free Electrons at ELC
- Linux kernel 2.6.29 - New features for embedded users
- The Buildroot project begins a new life
- FOSDEM 2009 videos
- USB-Ethernet device for Linux
- Program for Embedded Linux Conference 2009 announced
- Public session changes
- Real hardware in our training sessions
- Call for presentations for the LSM embedded track

Docs

Most of the below documents are presentations used in our training sessions, or in technical conferences.

License

All our documents are available under the terms of the Creative Commons Attribution-ShareAlike 3.0 license. This essentially means that you are free to download, distribute and even modify them, provided you mention us as the original authors and that you share these documents under the same conditions.

Linux kernel

- Embedded Linux kernel and driver development
- New features in Linux 2.6 (since 2.6.10)
- Kernel initialization
- Porting Linux to new hardware
- Power management in Linux
- Linux PCI drivers
- Block device drivers
- Linux USB drivers
- DMA

Architecture specific documents

- ARM Linux specifics
- Linux on TI OMAP processors

Embedded Linux system development

- Embedded Linux system development
- Real time in embedded Linux systems
- Block filesystems
- Flash filesystems
- Free software development tools
- The U-boot bootloader
- The GRUB bootloader
- The blob bootloader
- Hotplugging with udev
- Introduction to uClinux
- Java in embedded Linux
- Embedded Linux optimizations
- Audio in embedded Linux systems
- Multimedia in embedded Linux systems
- Embedded Linux From Scratch... in 40 minutes!
- Building embedded Linux systems with Buildroot
- Developing embedded distributions with OpenEmbedded
- The Scratchbox development environment

Miscellaneous

- Introduction to the Unix command line
- SSH
- Linux virtualization solutions (with an embedded perspective)
- Advantages of Free Software and Open Source in embedded systems
- Introduction to GNU/Linux and Free Software

All our technical presentations on http://free-electrons.com/docs

- Linux kernel
- Device drivers
- Architecture specifics
- Embedded Linux system development

How to help

You can help us to improve and maintain this document...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order development, consulting and training services performed by the authors of these documents (see http://free-electrons.com/).
- By sharing this document with your friends, colleagues and with the local Free Software community.
- By adding links on your website to our on-line materials, to increase their visibility in search engine results.

Free Electrons

Our services

Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Application and interface development

Consulting and technical support

Help in decision making
System architecture
System design and performance review
Development tool and application support
Investigating issues and fixing tool bugs

Linux kernel

Linux device drivers
Board support code
Mainstreaming kernel code
Kernel debugging

Embedded Linux Training

All materials released with a free license!

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux, uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization