The Plan

A quick review of the kernel development process
  How it works
  Current issues of interest

Recent history review
  What has happened over the last year

Looking forward
  Wild predictions about future kernels
The kernel release process

Major kernel releases about every 3 months
Named 2.6.x
2.6.x.y releases for important fixes
   Security problems
   System crashes

Every 2.6.x is a major release
   New features
   Internal API changes

Where's 2.7?
   The old even/odd scheme is no more
The kernel release lifecycle

Week 0: the merge window opens
   All new features and major changes merged
   Can be several thousand patches

Week 2: 2.6.x-rc1 is released
   Merge window closes – no new features (usually)
   Patch rate remains high – but should all be fixes

Weeks 3-8: additional -rc releases
   Patch rate slows as bugs get fixed

Week 8: 2.6.x is released
   2.6.x.y bug fix releases come later
The kernel release lifecycle

Week 0: the merge window opens
All new features and major changes merged
Can be several thousand patches

Week 2: 2.6.x-rc1 is released
Merge window closes – no new features (usually)
Patch rate remains high – but should all be fixes

Weeks 3-8: additional -rc releases
Patch rate slows as bugs get fixed

Week 8: 2.6.x is released
2.6.x.y bug fix releases come later
The patch rate

Patching Rate

9,200 lines changed per day
Some statistics

Since 2.6.16 (just over 1 year ago):
30182 changesets merged
2074 developers contributed to the kernel
10 contributed >= 1% of changes
766,000 lines added to the kernel

Who do they work for?

<table>
<thead>
<tr>
<th>Company</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>25%</td>
</tr>
<tr>
<td>Red Hat</td>
<td>14%</td>
</tr>
<tr>
<td>Volunteer</td>
<td>12%</td>
</tr>
<tr>
<td>IBM</td>
<td>8%</td>
</tr>
<tr>
<td>Novell</td>
<td>4%</td>
</tr>
<tr>
<td>Qlogic</td>
<td>4%</td>
</tr>
<tr>
<td>Intel</td>
<td>3%</td>
</tr>
<tr>
<td>MIPS Tech.</td>
<td>2%</td>
</tr>
<tr>
<td>MontaVista</td>
<td>2%</td>
</tr>
<tr>
<td>Nokia</td>
<td>2%</td>
</tr>
<tr>
<td>SANPeople</td>
<td>1%</td>
</tr>
<tr>
<td>SteelEye</td>
<td>1%</td>
</tr>
<tr>
<td>Freescale</td>
<td>1%</td>
</tr>
<tr>
<td>Simtec</td>
<td>1%</td>
</tr>
<tr>
<td>Astaro</td>
<td>1%</td>
</tr>
<tr>
<td>Linux Foundation</td>
<td>1%</td>
</tr>
<tr>
<td>Atmel</td>
<td>1%</td>
</tr>
<tr>
<td>Oracle</td>
<td>1%</td>
</tr>
<tr>
<td>HP</td>
<td>1%</td>
</tr>
<tr>
<td>SGI</td>
<td>1%</td>
</tr>
</tbody>
</table>
The results

The patch flow rate is high
   New features get to users more quickly
   Distributor kernels stay closer to the mainline

Relatively predictable kernel releases

Happy distributors, developers, and users
   ...most of the time
Kernel Quality

“I believe the 2.6 kernel is slowly getting buggier. It seems we're adding bugs at a higher rate than we're fixing them.”
-- Andrew Morton, May, 2006

Some fear that kernel quality is declining
  Bugs not getting fixed
  Too many features added too quickly
  Too little stabilization time

Kernel developers tend not to agree
  But everybody agrees fewer bugs would be better
A quick review of the last year

2.6.16 (March 19, 2006)
- Mutexes replace semaphores
- High-resolution timer code
- OCFS2 cluster filesystem
- SCHED_BATCH

2.6.17 (June 17, 2006)
- SPARC Niagara support
- Lightweight robust futexes
- User-space software suspend
- Broadcom 43xx wireless support
- splice()
Still reviewing last year

2.6.18 (September 19, 2006)
  Priority inheritance
  Generic IRQ layer
  New core time subsystem
  Kernel locking validator
  Devfs gone

2.6.19 (November 29, 2006)
  Parallel ATA driver subsystem
  GFS2 cluster filesystem
  ext4 development filesystem
  eCryptfs
The current kernel

2.6.20 (February 4, 2007)
  Fault injection framework
  Many big internal API changes
  UDP-Lite protocol
  paravirt_ops
  Kernel virtual machine (KVM)
  Playstation 3 support
Looking forward

Predicting the kernel's future is hard
  No five-year roadmaps
  No ability to force work from anybody
  No limits on what people might come up with

I won't let that stop me
  I can handwave with the best

How does one proceed?
  Look at work in progress now
  Look at pressures from the outside world
  Make some wild guesses

Woe to anybody who actually believes what follows...
The next kernel

2.6.21 (any day now)

What's going in?
- Dynamic tick and clockevents
- Major ACPI update
- Sysfs shadow directories
- ALSA system-on-chip layer
- Device resource management API
- VMI virtualization interface
- KVM improvements (live migration)
Virtualization

Still an area of high interest
Server consolidation
High-reliability systems
Isolation and security

The big players
Xen
  Full paravirtualization
  Path into the kernel has been slow
User-mode Linux
  Run Linux as a user-mode process
  Longstanding Linux project
Various commercial offerings

The biggest development issue:
A common hypervisor interface
Virtualization developments

paravirt_ops
- The common hypervisor interface
- Isolates low-level operations
- Run-time substitution via “hypervisor ROM”
- Remains a highly volatile interface

VMI
- Higher-level hypervisor interface

Kernel Virtual Machine
- Support for hardware virtualization
- Open /dev/kvm, create CPUs with ioctl(), launch systems
- A full virtualization solution
  ...but paravirtualization being done too

Lguest (aka Rustyvisor)
- A simple native Linux virtualization mechanism
Containers

A lighter-weight approach to virtualization

No full emulation of the processor
  Containers run as process groups on host
  All containers run on the host kernel

Containers are isolated from each other
  Can't see other processes
Containers

There are a number of container projects
  Linux-VServer
  OpenVZ
  Various proprietary offerings

All have the same needs
  Multiple views of global resources
  Per-container resource usage control

Most of them want into the kernel
  But multiple implementations are unwelcome

The projects are talking to each other
  Some early code bits have been merged
  Big issues: resource management, networking, ...
CPU schedulers

Scheduling has been quiet for some time
Worst problems solved in early 2.6.x

The issue has come back
Better interactive response wanted
Dump complex heuristics for simple fairness

Three contenders
Staircase Deadline
Completely Fair Scheduler
Nicksched

CFS looks to be the likely winner
...but expect some debate first
Asynchronous I/O is a perennial pain
State-machine approach difficult to implement, maintain

Fibrils: a new approach
If something blocks, keep running in a new process
Makes *any* system call asynchronous

Syslets
Variant of fibrils
Applications can load code into the kernel

Threadlets
On-demand threading
Simple API
Filesystems

Pressures
Disks are getting bigger – quickly
They are getting faster much less quickly
The time to read the entire disk is growing
They are not getting more reliable
Some filesystem limits are being reached

How long does it take to run fsck?
Kernel.org RAID: over 1 week

Current filesystems have a long history

“We're continuing to nurse along a few basically-15-year-old filesystems while we do have the brains, manpower, and processes to implement a new, really great one.”
--Andrew Morton
Filesystems – what's coming

ext4
Currently a development-only filesystem
Extents
48-bit block numbers (break the 8TB limit)

Reiser4
A number of interesting new ideas
Still stalled – won't be in 2.6.22 either
Future is now in serious doubt
Hardware support

Hardware support is better than ever
Most hardware Just Works
No driver disks, no hassles
Linux supports more hardware than any other system, ever

There are exceptions
Wireless networking
Video adapters

The problem
Vendors will not release free drivers
...or programming information
Why not release information?

“It's so hard to write a graphics driver that open-sourcing it would not help.”
-- Andrew Fear, Nvidia software product manager

Other issues
Patent problems
Regulatory issues
They just plain don't get it
Wireless networking

Wireless has traditionally been poorly supported
  Few drivers
  Suboptimal network stack design

The mac80211 (formerly Devicescape) stack
  A proper 802.11 networking stack
  Slowly making its way toward the mainline

New drivers
  Broadcom 43xx
  Atheros
    Now cleared of legal clouds
  Intel
    Well supported by the vendor
Video adapters

Video vendors remain stubborn

Intel the biggest exception
Still short on programming information
Integrated controllers only – for now

Nvidia
The Nouveau project is moving forward
nouveau.freedesktop.org
Some ground to cover yet

ATI
R300 driver is getting good
Little hope for newer chipsets
Binary-only drivers

Some vendors do provide proprietary drivers

Some problems:
- Only work with specific kernel versions
- Unknown security problems
- No hope for fixing bugs
- No support for other architectures
- Long-term support is dubious
- Can impede development
- Questionable legality

Linux cannot give in to binary-only drivers
That way leads to the end of our free system
Networking

Network channels
Presented by Van Jacobson at Ica 2006
Push network processing to the end points
...even into user space
Progress is slow

Needed: an event reporting API
Unify application event loops
Improve high-bandwidth application performance

The new eventfd system calls:
Get a file descriptor for interesting events
Timers, signals, etc.
Wait for them in the poll() loop

The kevent mechanism
Seemingly superseded by eventfd
Security

SELinux: The one true security framework?
- Becoming more comprehensive (packet labeling)
- Higher-level admin tools

AppArmor
- Pushed by Novell/SUSE
- Much simpler administration
- Unpopular with developers – use of pathnames
- New patch set posted (finally)

SLIM, EVM, and friends
- Use the TPM for integrity management
- Can be used for high security – or lockdown
- Slow path into kernel
Real time

The realtime preemption patch set
Claims 20 μsec deterministic response time
Large invasive patch set

Much of it has already been merged
Robust futexes, priority inheritance, mutexes
core timekeeping, high-resolution timers, ...

Some pieces remain
Sleeping spinlocks
Interrupt handlers in kernel threads
Dynamic tick
Small and embedded systems

Much is happening in small systems
  Telephones
  Tablet systems
  OLPC

Running Linux there presents different challenges
  Minimal resource use
  Real-time response
  Fast boot

Lots of people are working in this area
  But cooperation is often lacking
  Little participation in the development process
    Proprietary hardware
  Things are getting better – maybe
Licensing and GPLv3

Version 3 of the GPL is still in draft form
Final version due in June

Relatively unpopular in kernel circles
The anti-DRM provisions in particular

The kernel is explicitly licensed under GPLv2
The “or any later version” language is missing

Changing the license would be hard
Hundreds of copyright holders
Achieving a consensus is unlikely
Even finding them all would be hard

Thus:
A GPLv3-licensed kernel is unlikely
Questions?

Slides at http://lwn.net/talks/elc2007/
The user-space API

The user-space API used to be simple
System calls

Now it is more complicated
Lots of system calls
/proc (100's of files)
/sys (1000's of files)
Netlink

Breaking this API is against the rules

But it is happening anyway
Such a wide interface is easy to break
Sysfs directly mirrors internal data structures
These APIs are still evolving
Scalability

Today's big iron is tomorrow's laptop
Supporting 1GB of memory was once a big deal

The current state of the art
512-processor NUMA systems work well
  Getting larger
  Getting to 4K will take some work

The scalability effort continues
Shrinking data structures
Lockless algorithms
...

...
Questions to ask

Is there really a problem?

If so, what is to be done about it?
What to do about it?

Regardless of whether kernel bugs are getting worse
...it would be nice to have fewer of them

More testing is needed
   By users!

Better bug tracking
   Special tracking for regressions

Better bug fixing
   Fixing bugs can be hard work
      No access to the hardware – unable to reproduce the problem
   Developer discipline can be lacking
   Known bugs often remain unfixed.
What to do about it?

Make bugs harder to introduce
  Better internal APIs
  Better automated tools
    Locking validator
    Sparse
    Fault injection framework
    Memory leak tracker

Stabilization releases
  Reserve occasional 2.6.x releases for bug fixes
  Seems to be a hard sell