LTSI Project Update

LTSI Kernel, How We Can Help Automotive Industries

Hisao Munakata, Tsugikazu Shibata

Linux Foundation Consumer Electronics working group

July 1st 2014

Who am I ? (Munakata)

- From embedded SoC provider company Renesas
- Linux Foundation CE[1] working Gr. Steering committee member, LF/CEWG Architecture Gr. co-chair
- One of LF/CEWG LTSI[2] project initial proposer
- At my company, I had been encouraging my team developers to send a patches upstream
- Also I have supported various CE customers who develop digital-TV, Blu-ray recorder and Smart-phone

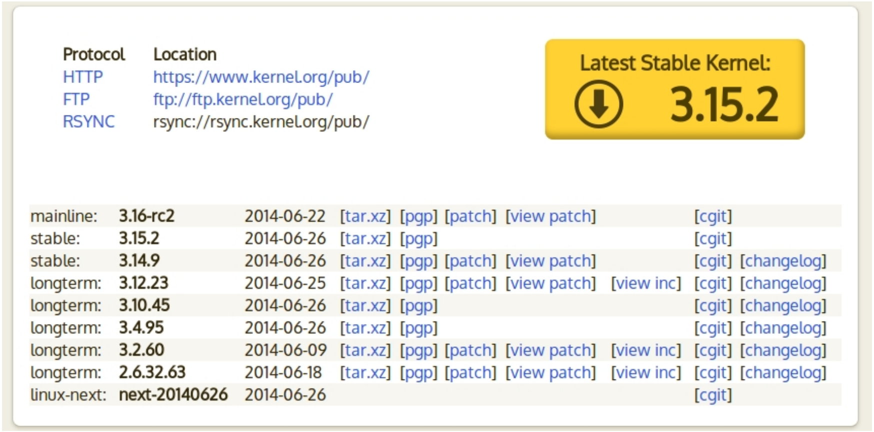
[1] CE = consumer electronics
[2] LTSI =Long Term Support Initiative

LTSI kernel update @ February 24, 2014

LTSI 3.0.79 --> 3.0.101 (EOL)
LTSI 3.4.46 --> 3.4.81 (update)

Why you should consider adopting LTS ?

Upstream kernel @kernel.org



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

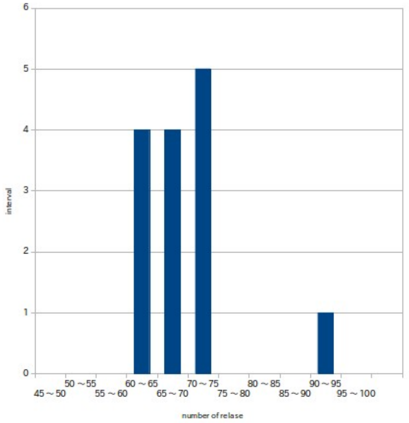
Latest Stable Kernel:
3.15.2

mainline:	3.16-rc2	2014-06-22	[tar.xz]	[pgp]	[patch]	[view patch]		[cgit]	
stable:	3.15.2	2014-06-26	[tar.xz]	[pgp]				[cgit]	
stable:	3.14.9	2014-06-26	[tar.xz]	[pgp]	[patch]	[view patch]		[cgit]	[changelog]
longterm:	3.12.23	2014-06-25	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit]	[changelog]
longterm:	3.10.45	2014-06-26	[tar.xz]	[pgp]				[cgit]	[changelog]
longterm:	3.4.95	2014-06-26	[tar.xz]	[pgp]				[cgit]	[changelog]
longterm:	3.2.60	2014-06-09	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit]	[changelog]
longterm:	2.6.32.63	2014-06-18	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit]	[changelog]
linux-next:	next-20140626	2014-06-26						[cgit]	

You can find 1)latest released, 2)under development (=mainline, next), and several stable kernels

The release record of 3.0 series upstream kernel

version	release date	duration
v3.1	2011-10-24	95 days
v3.2	2012-01-04	72 days
v3.3	2012-03-18	74 days
v3.4	2012-05-20	63 days
v3.5	2012-07-21	62 days
v3.6	2012-09-30	71 days
v3.7	2012-12-10	71 days
v3.8	2012-02-18	70 days
v3.9	2013-04-28	69 days
v3.10	2013-06-30	63 days
v3.11	2013-09-02	64 days
v3.12	2013-11-15	74 days
v3.13	2014-01-21	67 days
v3.14	2014-03-30	68 days
v3.15	2014-06-08	70 days



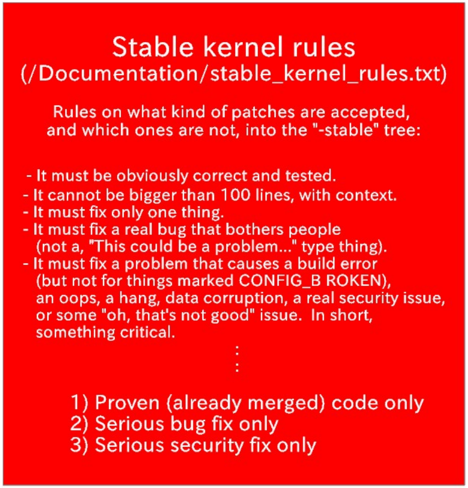
Release happened regularly at around every 70 days

However, not all kernels are maintained for longterm

version	maintenance status
v3.1	maintained till 3.1.9, then now EOL
v3.2	longterm (3.2.55), kept maintained (by Debian)
v3.3	maintained till 3.3.8, then now EOL
v3.4	longterm (3.4.95), kept maintained
v3.5	maintained till 3.5.7, then now EOL
v3.6	maintained till 3.6.11, then now EOL
v3.7	maintained till 3.7.10, then now EOL
v3.8	maintained till 3.8.13, then now EOL
v3.9	maintained till 3.9.11, then now EOL
v3.10	longterm stable (3.10.45), kept maintained
v3.11	maintained till 3.11.10, then now EOL
v3.12	longterm stable (3.12.23), kept maintained (by ???)
v3.13	stable release (3.13.11), till 3.15 released
v3.14	next longterm stable version (3.14.9)
v3.15	latest release (3.15.2), will be maintained as stable till 3.17 is out

Stable release include MUST APPLY essential fixes

version	fixes
v3.0 -> v3.0.101	3,953
v3.1 -> v3.1.9	647
v3.2 -> v3.2.60	5,001
v3.3 -> v3.3.8	698
v3.4 -> v3.4.95	4,506
v3.5 -> v3.5.7	816
v3.6 -> v3.6.9	676
v3.7 -> v3.7.10	718
v3.8 -> v3.8.13	996
v3.9 -> v3.9.11	746
v3.10 -> v3.10.45	2,970
v3.11 -> v3.11.10	677
v3.12 -> v3.12.24	2,314
v3.13 -> v3.13.11	903
v3.14 -> v3.14.9	845



Stable kernel rules
(/Documentation/stable_kernel_rules.txt)

Rules on what kind of patches are accepted,
and which ones are not, into the "-stable" tree:

- It must be obviously correct and tested.
- It cannot be bigger than 100 lines, with context.
- It must fix only one thing.
- It must fix a real bug that bothers people
  (not a, "This could be a problem..." type thing).
- It must fix a problem that causes a build error
  (but not for things marked CONFIG_B ROKEN),
  an oops, a hang, data corruption, a real security issue,
  or some "oh, that's not good" issue. In short,
  something critical.

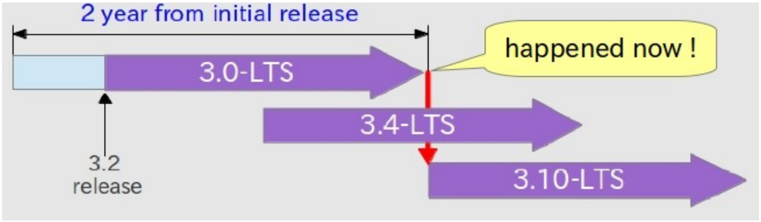
⋮

1) Proven (already merged) code only
2) Serious bug fix only
3) Serious security fix only

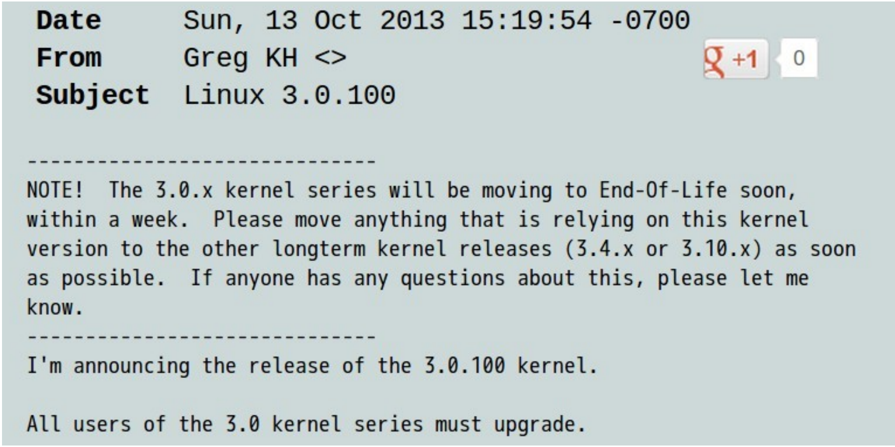
Longterm stable (LTS) kernel release cadence

Target kernel selection rules

- Maintainer will **choose one LTS version per year**
- **Maintain it for 2 years** from its original release
- LTSI-3.0 is moved to EOL when 3.10 became new LTS
- Then, we have 2 LTS kernels versions like 3.4 and 3.10



LTS/LTSI maintainer, Greg says 3.0 moves to EOL



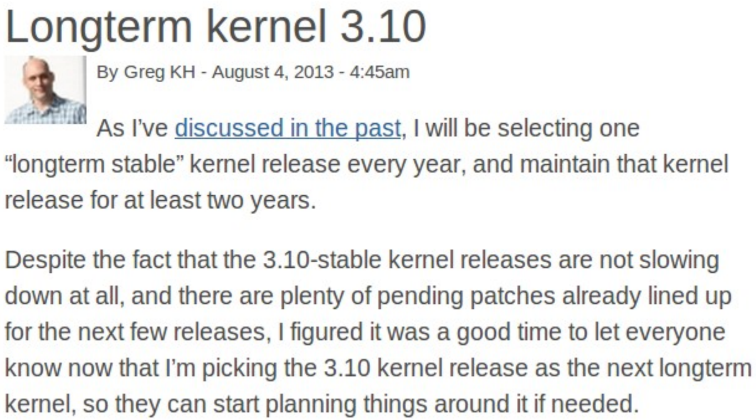
Date	Sun, 13 Oct 2013 15:19:54 -0700
From	Greg KH <>
Subject	Linux 3.0.100

```
------------------------------
NOTE!  The 3.0.x kernel series will be moving to End-Of-Life soon,
within a week.  Please move anything that is relying on this kernel
version to the other longterm kernel releases (3.4.x or 3.10.x) as soon
as possible.  If anyone has any questions about this, please let me
know.
------------------------------
I'm announcing the release of the 3.0.100 kernel.

All users of the 3.0 kernel series must upgrade.
```

https://lkml.org/lkml/2013/10/13/160

Greg also announced longterm for 2014 is 3.10



Longterm kernel 3.10

By Greg KH - August 4, 2013 - 4:45am

As I've discussed in the past, I will be selecting one "longterm stable" kernel release every year, and maintain that kernel release for at least two years.

Despite the fact that the 3.10-stable kernel releases are not slowing down at all, and there are plenty of pending patches already lined up for the next few releases, I figured it was a good time to let everyone know now that I'm picking the 3.10 kernel release as the next longterm kernel, so they can start planning things around it if needed.

http://www.linuxfoundation.org/news-media/blogs/browse/2013/08/longterm-kernel-310

Upstream kernel 3.10 development (is done)

Whoops, we can not submit our latest device support code to 3.0 kernel now! Yes, that is true, because

item	date
kernel 3.10 merge window open	2013.4.28
kernel 3.10 merge window close	2013.5.12
kernel 3.10 release	2013.6.30

As upstream 3.10 patch merge window is already closed, there is no chance to add your code to upstream kernel. Thus a cutting-edge silicon release after development cycle can not be supported in longterm 3.10 kernel. This might be problematic for embedded industry Linux adopter.

LTSI 3.10 development result review

LTSI-3.10 development history

item	date
kernel 3.10 merge window open	2013.4.28
kernel 3.10 merge window close	2013.5.12
kernel 3.10 release	2013.6.30
Announce of 2013 LTS kernel version	2013.8.4
LTSI-3.10 git tree open	2013.9.11
3.10 becomes LTS (=3.12 release)	2013.11.15
LTSI-3.10 merge window open	2013.11.15
patch collection period	75 days
LTSI-3.10-rc1 (=merge window close)	2014.1.29
validation period	26 days
LTSI-3.10 release	2014.2.24

Major contributor for LTSI-3.10

Contributor	Patch count
Darren Hart (intel)	1,197
Simon Horman (for Renesas)	1,122
Daniel Sangorrin (Toshiba)	123
Patrik Jakobsson (for intel)	46
Mark Brown (linaro.org)	11
Greg Kroah-Hartman (Linuxfoundation)	11
Total	2,510

Major achievement of LTSI 3.10

- LTTng
- Power efficient workqueues
- Intel's BayTrail support
- Intel's Minnowboard support
- Renesas's R-Car H2/M2 series support backported from the latest mainline
- Xilinx Zynq board support

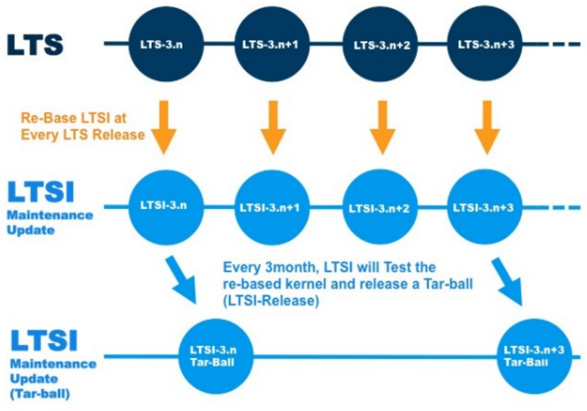
Hot news ! Greg announced 2014 LTS will be 3.14

At the ELC2014 conference LTSI workshop, Greg stated next LTS (and LTSI) kernel version would be 3.14.

item	date
kernel 3.14 merge window open	2014.1.9
kernel 3.14 merge window close	2014.2.2
kernel 3.14 release	2014.3.30
LTSI-3.14 merge window open (target)	2014.8.21
patch collection period	70 days
LTSI-3.14-rc1 (=merge window close, target)	2014.10.30
validation period	50+ days
LTSI-3.14 release (target)	2014.12.25?

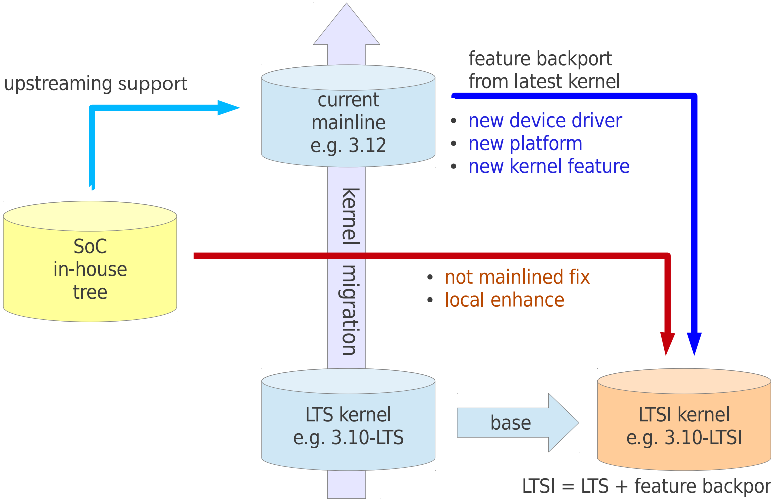
Please be ready for collecting patches to send LTSI-3.14 now!

Hot news 2! New LTS to LTSI update reflection cycle

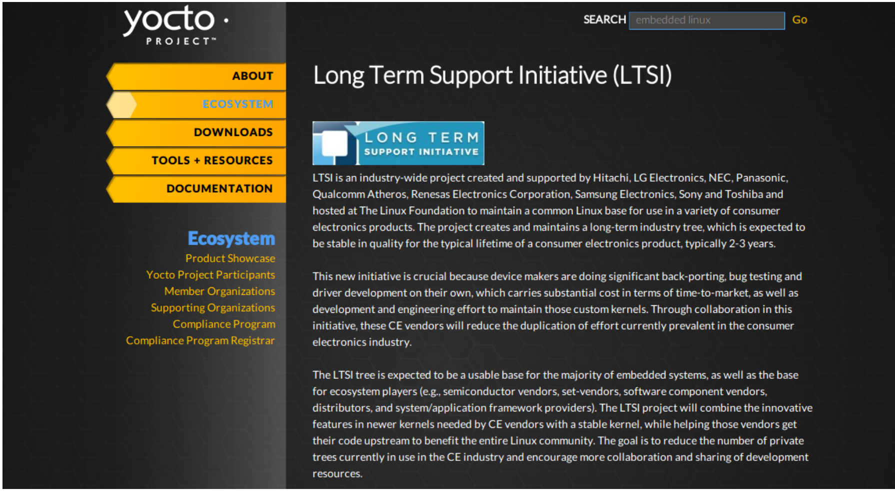

LTS
LTS-3.n
LTS-3.n+1
LTS-3.n+2
LTS-3.n+3
Re-Base LTSI at Every LTS Release
LTSI Maintenance Update
LTSI-3.n
LTSI-3.n+1
LTSI-3.n+2
LTSI-3.n+3
Every 3month, LTSI will Test the re-based kernel and release a Tar-ball (LTSI-Release)
LTSI Maintenance Update (Tar-ball)
LTSI-3.n Tar-Ball
LTSI-3.n+3 Tar-Ball

Every stable update will be ported to existing LTSI code

LTSI = community LTS(longterm) + industry extra



Yocto and LTSI project coordination is working now



Long Term Support Initiative (LTSI)

LTSI is an industry-wide project created and supported by Hitachi, LG Electronics, NEC, Panasonic, Qualcomm Atheros, Renesas Electronics Corporation, Samsung Electronics, Sony and Toshiba and hosted at The Linux Foundation to maintain a common Linux base for use in a variety of consumer electronics products. The project creates and maintains a long-term industry tree, which is expected to be stable in quality for the typical lifetime of a consumer electronics product, typically 2-3 years.

This new initiative is crucial because device makers are doing significant back-porting, bug testing and driver development on their own, which carries substantial cost in terms of time-to-market, as well as development and engineering effort to maintain those custom kernels. Through collaboration in this initiative, these CE vendors will reduce the duplication of effort currently prevalent in the consumer electronics industry.

The LTSI tree is expected to be a usable base for the majority of embedded systems, as well as the base for ecosystem players (e.g., semiconductor vendors, set-vendors, software component vendors, distributors, and system/application framework providers). The LTSI project will combine the innovative features in newer kernels needed by CE vendors with a stable kernel, while helping those vendors get their code upstream to benefit the entire Linux community. The goal is to reduce the number of private trees currently in use in the CE industry and encourage more collaboration and sharing of development resources.

Discipline of LTSI project management

- Community LTS + industry demanded extra patches.
- Governed by LF/CEWG
- Focus on kernel code[a], not aiming complete BSP
- Therefore, can be combined with existing platform[b]
- CPU architecture and platform neutral
- Comply with upstream rules[c]
- Industry friendly acceptance (flexible patch forms, etc)
- Help CE (and others) industry to utilize Linux

[a] device drivers are part of kernel, of course
[b] Android, Yocto, Tizen, AGL, WebOS and others
[c] e.g. signed-off-by process

Shared (LTSI) kernel test project

Why LTSI kernel validation becomes important ?

- Upstream LTS is managed to be completely safe.
- LTSI can based on community LTS kernel, and
- LTSI is the place to add various NEW things
  - Feature back port from latest mainline (relatively safe)
  - Industry demanded not-mainlined (but commonly used) open source project code
  - Privately maintained bug-fix code (may be valuable)
  - Privately developed feature code

We want to validate LTSI kernel does not include any bug or regression against the community LTS code

Beyond the LTS(I) kernel use, share the test case !

New value opportunity of sharing the kernel test case

- Now many industry start using LTS and LTSI kernel.
- Each company may spend a lot of time for validation.
- Some of fundamental kernel feature test might be duplicated
  - common kernel function test (detail later)
  - common kernel benchmark test (detail later)
  - common compatibility conformance test
- Now we can consider sharing the (part of) kernel test case on top of LTS(I) kernel across the industry.
- We need to assign appropriate OSS license to each test case itself so the we can share them.

Design target of shared LTSI test environment

feature

- Fully automated execution (nightly run)
- Easy to manage operation (add/edit test case)
- Trend monitoring capability (to catch the regression)
- User friendly interface (web access, GUI front end)

operation

- local text execution (can install to your computer)
- test case sharing mechanism
- test result sharing mechanism (future work)
- can penetrate to the upstream kernel development use

Current shape

We did trial run during LTSI-3.10 development period

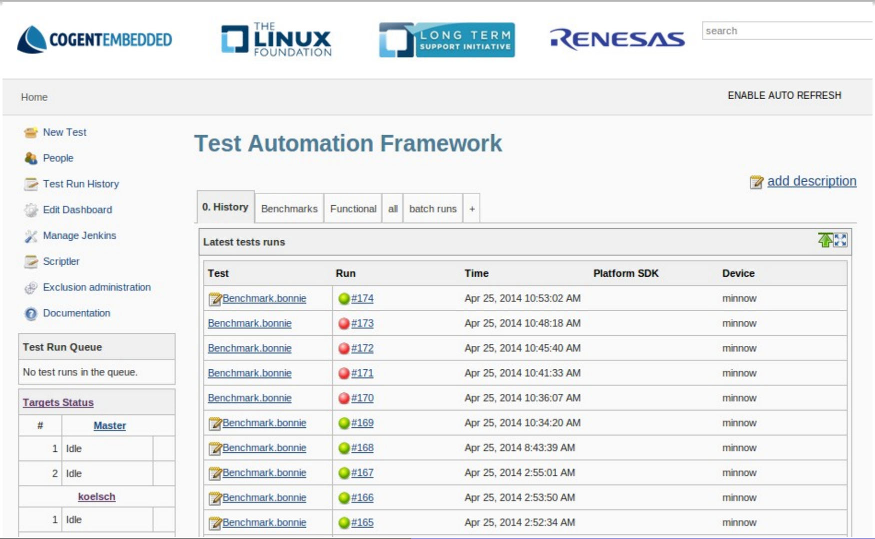
- Cogent Embedded / Renesas worked together.
- We will donate environment to public so that anyone can execute pre-build test and write own test case.

- Jenkins front-end (test automation)
- Customizations (UI/look&feel, representation)
- Open Source Test Suites (public, popular) are integrated
- Some private test suites (shell scripts) tested

Implementation

- Jenkins brings - mature, robust platform to manage & distribute jobs (could be test/tests suites distributed across various platforms)
- Tests/test suites wrapped into shell scripts. Idea is to keep environment as simple/straightforward as possible:
  - Every step is a script: build test, deploy on target, run, collect results, parse results, cleanup
  - It should be possible to trigger scripts, run tests, collect results without complex Jenkins setup
- Targets are connected with server(s) via network (e.g. debug ethernet) and/or serial
- Test results, status, statistics, Target configuration, etc - visualized by Jenkins (accessible via web interface)

Screen shot (Jenkins Web based test controller UI)



Test Automation Framework

Test Run Queue

No test runs in the queue.

Targets Status

#	Master
1	Idle
2	Idle
	koelsch
1	Idle

Latest tests runs

Test	Run	Time	Platform SDK	Device
Benchmark.bonnie	🟢 #174	Apr 25, 2014 10:53:02 AM		minnow
Benchmark.bonnie	🔴 #173	Apr 25, 2014 10:48:18 AM		minnow
Benchmark.bonnie	🔴 #172	Apr 25, 2014 10:45:40 AM		minnow
Benchmark.bonnie	🔴 #171	Apr 25, 2014 10:41:33 AM		minnow
Benchmark.bonnie	🔴 #170	Apr 25, 2014 10:36:07 AM		minnow
Benchmark.bonnie	🟢 #169	Apr 25, 2014 10:34:20 AM		minnow
Benchmark.bonnie	🟢 #168	Apr 25, 2014 8:43:39 AM		minnow
Benchmark.bonnie	🟢 #167	Apr 25, 2014 2:55:01 AM		minnow
Benchmark.bonnie	🟢 #166	Apr 25, 2014 2:53:50 AM		minnow
Benchmark.bonnie	🟢 #165	Apr 25, 2014 2:52:34 AM		minnow

Currently integrated 28 automated benchmarks

1 aim7
2 blobsallad
3 bonnie
4 Dhrystone
5 cyclictest
6 fio
7 GLMark
8 ebizzy
9 ffsb
10 hackbench
11 gtkperf
12 himeno
13 Interbench
14 IOzone
15 iperf
16 Java
17 linpack
18 lmbench2
19 nbench-byte
20 netperf
21 netpipe
22 OpenSSL
23 reboot
24 Stream
25 signaltest
26 tiobench
27 Whetstone
28 x11perf

Currently integrated 33 automated tests

1	aiostress	12	LTP-DDT.Devices	23	netperf
2	bzip2	13	LTP-DDT.Ipc	24	OpenSSL
3	expat	14	LTP-DDT.Math	25	pi_tests
4	cmt_RENESAS	15	LTP-DDT.Mm	26	posixtestsuite
5	crashme	16	LTP-DDT.Nptl	27	rmaptest
6	ipv6connect	17	LTP-DDT.Pipes	28	scifab_RENESAS
7	fontconfig	18	LTP-DDT.Syscalls	29	scrashme
8	ft2demos	19	LTP-DDT.Timers	30	sdhi_0_RENESAS
9	glib	20	LTP.Devices	31	stress
10	jpeg	21	LTP.Filesystem	32	synctest
11	linus_stress	22	LTP.Open_Posix	33	zlib

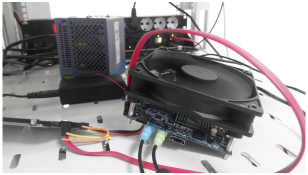
You can integrate your own test case (public/private) here

Target configuration

- Target abstraction is just a set of environment variables
  - Target Architecture (ARM, x86, MIPS) / toolchain path
  - IP addr/login pair if target is connected/controlled via TCP/IP (SSH, FTP, telnet) or serial port parameters
  - Target power-cycle settings
  - Target Linux distro-specific settings (if any)
  - temporary folder for test suites/logs
  - command to grab system logs
- Target pre-setup required
  - Bootcode + kernel (under test) + minimal distro

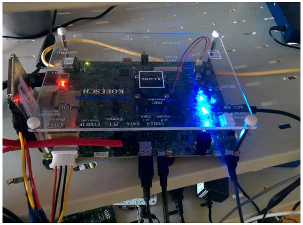
Result 1 : LTS 3.10 vs. LTSI-3.10

- Use Intel Atom Minnow board
- Minnow is a good platform to compare LTS and LTSI (at least works without additional patch lifting, headaches, etc) - example how mainline support should be done.
- No significant deviation in results observed, no major regressions.
- Anomaly found: fio-1.58 fails when running on mSD card using LTSI-3.10 (worked ok with LTS-3.10), on the other side, newer version of fio does not show similar anomaly.



Result 2 : LTSI-3.4 vs. LTSI-3.10

- Use Renesas R-CarM2 Koelsch board
- Renesas is using LTSI kernels as a baseline for product-quality BSPs delivered to customers. R-Car M2 - good candidate to compare LTSI3.4 and LTSI3.10 code bases.
- No significant deviation in results observed, no major regressions.
- Anomaly found: cyclic test fails (will study further/deeper soon)



Our struggles while we did LTSI test trial run

- We still don't have common understand/requirements
  - What we want to test and how we want to do this (need to collect feedback), example: some members mentioned - they don't see sense in testing on real hardware, some - want one feature tested (e.g. IPv6, using TAHI tests), some - another, etc. we need to have more formal approach to make testing useful.
- A number of problems when comparing tests results
  - Hardware support often behind, sometimes very different between various combination of linux kernels, etc. difficult to find hardware platform that would be well-maintained in LTS/LTSI trees for a while. Minnow - nice work, but now minnow-max is coming, etc.
  - Default configurations may get changes, behavior of some kernel features, etc. especially when comparing results from older release with new release.

New/extended auto-test functionality 1/2

- Report generation
  - framework allows to generate readable/standalone reports) - completed
- Documentation
  - Early version - completed.
  - Improvements - in progress.
- Integration/tuning of new tests
  - Renesas board-specific tests being added now
  - A few more open source tests suites (e.g. dbench, etc.)

New/extended auto-test functionality 2/2

Following features will be integrated soon.

- Serial port, ftp and telnet support
- ``board/target'' initialization/configuration/deployment
- Automated power cycle/reboot control integration
- Build everything from source
    - target kernel
    - bootcode
    - minimal distro
- UI/Jenkins plugins improvements

Conclusion

Conclusion

- Correct understanding of the longterm (LTS) cadence is important. LF/CEWG develops LTSI version on top of community longterm kernel. You can gain huge cost reduction if you can fully utilize LTS & LTSI scheme.
- 3.0 longterm maintenance cycle has been moved to EOL and3.10 is the next longterm support target. LTSI-3.10 was released in February 2014. And LTSI project maintainer Greg K.H.lately announced that next LTS(I) kernel will be 3.14.
- We have developed automated kernel test framework and tried with LTSI-3.10 release. We are hoping to share the kernel test case on top of commonly used LTSI kernel and upstream kernel development.

LONG TERM SUPPORT INITIATIVE

Resources

- project web = ltsi.linuxfoundation.org
- LTSI process document (new) = http://ltsi.linuxfoundation.org/participate-in-ltsi/ltsi-development-guide
- ML
  - ML subscription = https://lists.linuxfoundation.org/mailman/listinfo/ltsi-dev
  - ML archives = http://lists.linuxfoundation.org/pipermail/ltsi-dev/
  - ML patchwork = https://patchwork.kernel.org/project/ltsi-dev/list/
- git(each patch) = http://git.linuxfoundation.org/?p=ltsi-kernel.git;a=summary
- download (tar ball) = http://ltsi.linuxfoundation.org/downloads/releases