Yet another long-term stable kernel

introduction of Linux Foundation project named "LTSI"

Hisao Munakata

Renesas Solutions Corp.

June 7th 2012, LinuxCon Japan

Agenda

1. Why we have initiated LTSI project
   - community long-term maintenance
   - why we needed LTSI
2. LTSI creation process
   - patch collection
   - validation and maintenance
3. Why we suggest you to use LTSI
   - value offering
4. Status Update
   - LTSI 3.0 status as of May/31
   - next step
   - resources
   - conclusion & call for action

who am I

- From embedded SoC provider company Renesas
- Linux Foundation CE working Gr. Steering committee member LF/CEWG Architecture Gr. co-chair
- One of LF/CEWG LTSI project initial proposer
- LinuxCon Japan (and others) steering committee
- At my company, I had been encouraging my team developers to send patch to the upstream
- We have sent hundreds of patch to the LTSI 3.0

Question : How did you choose your kernel version ?

1. No care : part of distribution[a]
2. No care : part of application platform[b]

---

3. No option : development started with SoC vendor's BSP

---

4. Selected : selected previously adopted kernel
5. Selected : selected today's latest fresh kernel[c]

---

[a] i.e. Debian/Ubuntu/Fedora/...
[b] i.e. Android/Tizen/Genivi/...
[c] This is always our preference anyway

Vintage kernel is not always matured as it is not a wine

Software lifetime

Software will die when its maintenance becomes discontinued

- modern system software works as a part of connected world.
- application program and data changes (expands) all the time.
- So software need to be maintained throughout its lifetime.
  (Bug-fix, security-fix patch must be collected and applied.)

Some people want to stick on old kernel (like 2.6 series so far), as they think it is enough stabilized. However code maturity is fully depends on how actively that version is kept maintained. In other word, how many patch flow happening against that kernel.

Upstream kernel maintenance expiration date

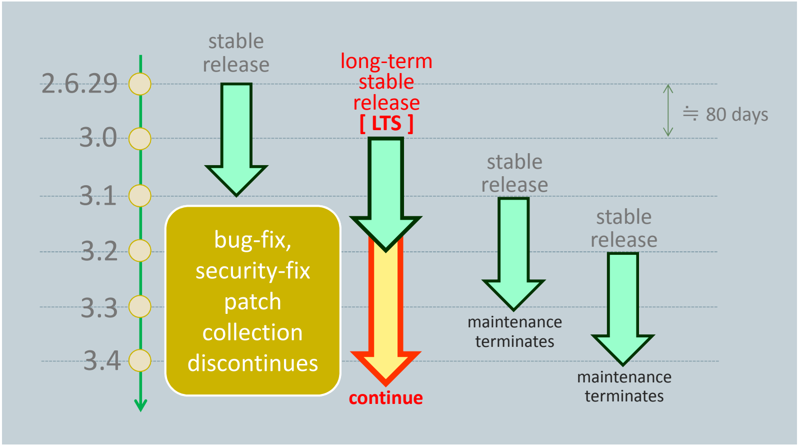
Maintenance period = accept bug-fix and/or security-fix patches. Once expired, community will discontinue such patch collection activity. If you want to keep it maintained after expiration, you need to invest someone to continue work on maintenance operation.

regular	kernel	=	roughly 80 days after its release
LTS[a]	kernel	=	normally a few year length

[a] long term stable version kernel

Selecting LTS version kernel should be the best practice to minimize long-term kernel maintenance cost.

Community maintenance : stable and long-term stable



LTS version selection criteria

LTS selection rules

- Bug-fix/security-fix patches must be provided to the enterprise server system for 5 years (or longer).
- Then Linux distro requested community to continue patch collection for specific version kernel.
- LTS version = 2.6.16 / 2.6.27 / 2.6.32 / 2.6.34 / 2.6.35. . .

New stable-kernel rule proposed by Greg Kroah-Hartman

- CE industry demand for LTS is different time-line
- select one version per year for LTS maintenance target
- keep it maintained for 2 year length after its release

OK, LTS seems a good choice, but...

Development speed of consumer product is relatively faster than other industry like enterprise, automotive world. This might cause delay of mainline kernel support catch up even such proposals are already submitted in upstream community.

timing gap challenge for early adoption

- Newly mainlined platform/SoC support is not available on LTS

Then, we envisioned to create LTSI to solve these embedded industry issues as an extension of community LTS release.

OK, LTS seems a good choice, but...

Development speed of consumer product is relatively faster than other industry like enterprise, automotive world. This might cause delay of mainline kernel support catch up even such proposals are already submitted in upstream community.

timing gap challenge for early adoption

- Newly mainlined platform/SoC support is not available on LTS
- Newly mainlined device drivers are still missing on LTS

Then, we envisioned to create LTSI to solve these embedded industry issues as an extension of community LTS release.

OK, LTS seems a good choice, but...

Development speed of consumer product is relatively faster than other industry like enterprise, automotive world. This might cause delay of mainline kernel support catch up even such proposals are already submitted in upstream community.

timing gap challenge for early adoption

- Newly mainlined platform/SoC support is not available on LTS
- Newly mainlined device drivers are still missing on LTS
- Newly mainlined kernel features are still not available on LTS

Then, we envisioned to create LTSI to solve these embedded industry issues as an extension of community LTS release.

OK, LTS seems a good choice, but...

Development speed of consumer product is relatively faster than other industry like enterprise, automotive world. This might cause delay of mainline kernel support catch up even such proposals are already submitted in upstream community.

timing gap challenge for early adoption

- Newly mainlined platform/SoC support is not available on LTS
- Newly mainlined device drivers are still missing on LTS
- Newly mainlined kernel features are still not available on LTS
- LTS kernel might not include device errata fix local patch

Then, we envisioned to create LTSI to solve these embedded industry issues as an extension of community LTS release.

LTS vs LTSI : What differs ?

LF/CEWG LTSI kernel

- **kernel features back-port form latest mainline**

community LTS kernel (is designed to be conservative)

- only accept bug-fix back-port
- only accept security-fix back-port

upstream kernel

- regularly migrated community kernel

LTS vs LTSI : What differs ?

LF/CEWG LTSI kernel

- **kernel features back-port form latest mainline**
- **device drivers back-port from latest mainline**

community LTS kernel (is designed to be conservative)

- only accept bug-fix back-port
- only accept security-fix back-port

upstream kernel

- regularly migrated community kernel

LTS vs LTSI : What differs ?

LF/CEWG LTSI kernel

- **kernel features back-port form latest mainline**
- **device drivers back-port from latest mainline**
- **local patch (=not yet mainlined) integration**

community LTS kernel (is designed to be conservative)

- only accept bug-fix back-port
- only accept security-fix back-port

upstream kernel

- regularly migrated community kernel

Concept of LTSI

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

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

Who can do what for LTSI

action	CEWG AG voting	CEWG Corporate	CEWG Individual	Other (public)
	membership category			
project web access	yes	yes	yes	yes
code download	yes	yes	yes	yes
ML subscription	yes	yes	yes	yes
attend ICM[1] meeting	yes	yes		
send patch proposal	yes	yes	yes	yes
review patch by ML	yes	yes	yes	yes
discuss patch at meeting	yes	yes	yes	
vote for patch (if needed)	yes			

If you plan to adopt LTSI to your product, we recommend you to join Linux Foundation and become a CEWG/AG member to get full advantage of LTSI project.

[1]industry contact meeting

Which code can be integrated to LTSI and how (1)

from upstream

newly adopted LTS bug-fix, security-fix	(automatic)
newly mainlined new kernel feature	(request basis)
newly mainlined new kernel driver	(request basis)
newly queued (to -rc version) code	(request basis)

from SoC vendor

curved up from SoC vendor kernel tree	(request basis)
submitted to upstream but still in review	(request basis)
not mainlined chip workaround code	(request basis)

Which code can be integrated to LTSI and how (2)

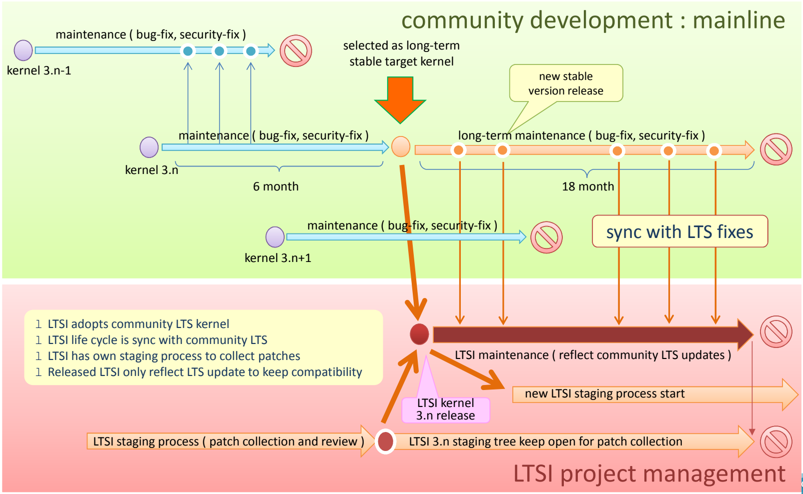
from product producer, distribution

- in-house bug-fix patch (request basis)
- private enhancement (request basis)
  - may require strict compatibility review

Others

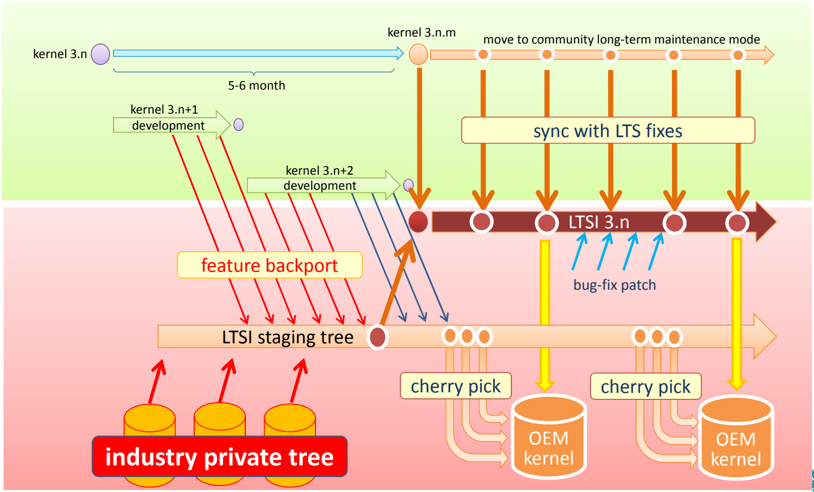
- not mainlined open-source project code (request basis)
  - ideally mainline attempt history already exist.
  - LTSI project will help re-attempt action.
- LF/CEWG funded open project result (request basis)

LTSI patch collection process


community development : mainline
maintenance (bug-fix, security-fix)
kernel 3.n-1
selected as long-term stable target kernel
new stable version release
maintenance (bug-fix, security-fix)
kernel 3.n
long-term maintenance (bug-fix, security-fix)
6 month
18 month
maintenance (bug-fix, security-fix)
kernel 3.n+1
sync with LTS fixes
LTSI maintenance (reflect community LTS updates)
new LTSI staging process start
LTSI kernel 3.n release
LTSI staging process (patch collection and review)
LTSI 3.n staging tree keep open for patch collection
LTSI project management

- LTSI adopts community LTS kernel
- LTSI life cycle is sync with community LTS
- LTSI has own staging process to collect patches
- Released LTSI only reflect LTS update to keep compatibility

LTSI maintenance process



LTSI staging process

Once patch collection window is closed, LTSI project will provide integrated snapshot for test that includes all approved patches.

During LTSI-staging period

- Anyone who send a patch should test your code.
- SoC vendor and distribution are expected to validate this snapshot using own reference development environment.
  - LTSI is CPU/platform neutral. You can use your environment.
  - You can publish test result like *"LTSI tested"* or *"LTSI ready"*
  - When you send a feedback, please clarify your environment (kernel config, test target spec ,reproduce procedure,..)
- If you hit any issues (bug, regression,..) please report it to ML. (We did not have dedicated BTS system like Bugzilla so far.)

Resolve SoC mainline support timing gap issue

SoC mainline support timing gap issue

Consumer product developer never (or, can not) wait until new SoC/Platform support code becomes part of mainline kernel.

- SoC companies are requested to deliver Linux BSP right after (within 0-3 month !) device sample release.
- Therefore SoC vendor needs to prioritize local in-house BSP development other than upstream patch submission.
- As a result, bunch of local in-house tree exit those hardly integrate to upstream, as its code manner is too unique.

LTSI can accept newly developed SoC support code to eliminate industry fragmentation (is in-house vendor tree).

[Example] R-mobileA1 *"Armaddilo"* platform support

We added new ARM platform support to LTSI 3.0

- Kernel 3.0 development completed July 2011
- Our new platform became ready in March 2012
- And We were requested to support kernel 3.0 with it
- Initially we submitted platform support patch to upstream[a]
- Once merged (or queued), then we wrote 3.4-rc to 3.0 backport patch set to add support in LTSI 3.0
- Finally *"Armadillo"* will be supported in both LTSI 3.0 and today's latest (and later) kernel

[a]We followed "upstream first" strategy as usual

LTSI can maximize your code reusability

Primarily KPI of software productivity is code reuse ratio

Linux kernel is originally designed to fit with various target with single configurable source.

- Apply collect abstraction (do not write immediate address)
- Follow the Linux standard framework (clock, DMA, GPIO,..) Otherwise your original framework will be overwritten.
- Isolate platform specific and generic code

LTSI project will help you to write community compatible reusable kernel code

LTSI milestone

2011.10.25	1st partner meeting@ELCE2011
2011.10.26	project launch @ELCE2011
2012.02.01	Greg Kroah-Hartman becomes LTSI chief maintainer
2012.02.14	2nd partner meeting@ELC2012
2012.02.16	3.0 staging-tree open
2012.02.17	project web page launched
2012.04.17	Japan workshop@Yokohama
2012.05.30	LTSI 3.0 merge window close
2012.06.08	1st Industry Contact Meeting [ICM]@Yokohama
2012.06.30	LTSI 3.0 code release
2012.08.29	2rd ICM @San Diego (planing)

LTSI ML posting status (as of end of May)

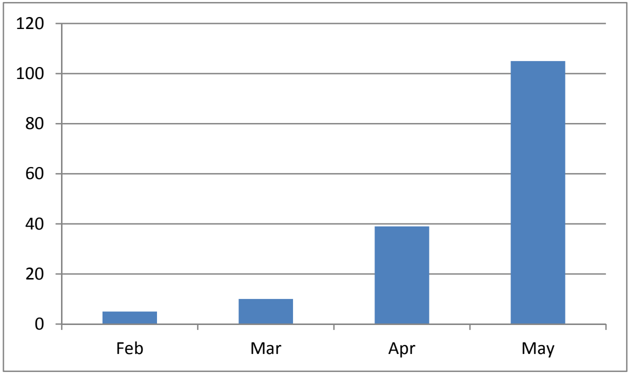


Figure: LTSI public_ML activity

Current LTSI tree status (as of end of May)

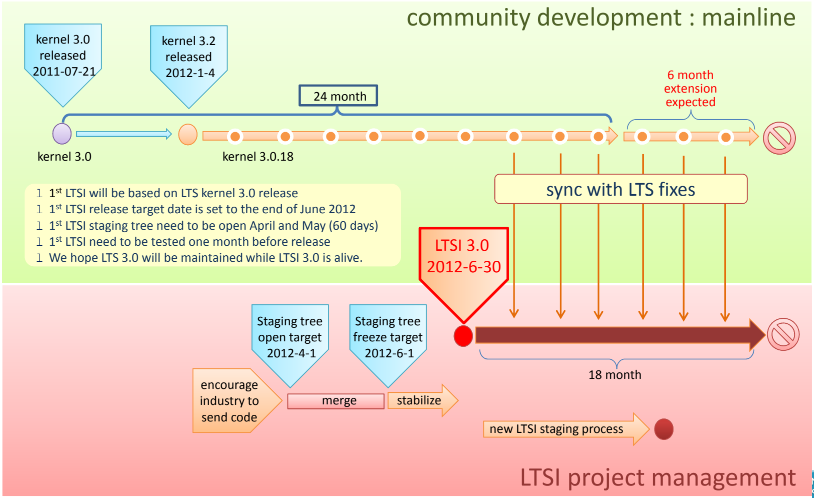
```
2012-05-17 23:35 Greg Kroah-Hartman o [master] [origin/HEAD][origin/master] fix up pramfs patch
2012-05-15 15:27 Greg Kroah-Hartman o pramfs patches added
2012-05-14 16:43 Greg Kroah-Hartman o update to 3.0.31
2012-05-14 16:38 Greg Kroah-Hartman o runtime pm patches added
2012-04-26 10:38 Greg Kroah-Hartman o Add armadillo800eva patches to the build
2012-04-25 17:38 Greg Kroah-Hartman o Update to latest version of LTTNG (v2.0.1)
2012-04-25 17:36 Greg Kroah-Hartman o Revert ‘‘remove old version of LTTNG’’
2012-04-25 17:35 Greg Kroah-Hartman o remove old version of LTTNG
2012-04-25 17:34 Greg Kroah-Hartman o update to 3.0.29
2012-02-15 15:33 Greg Kroah-Hartman o README: put proper ltsi-dev mailing list information in the file
2012-02-14 22:03 Greg Kroah-Hartman o initial scripts/generate_* script additions
2012-02-14 21:53 Greg Kroah-Hartman o create patches.ltsi for the ltsi project specific patches
2012-02-14 21:53 Greg Kroah-Hartman o README: added lots of info
2012-02-14 21:50 Greg Kroah-Hartman o KERNEL_VERSION added
2012-02-13 11:03 Greg Kroah-Hartman o add forgotten lttng patch
2012-02-12 23:23 Greg Kroah-Hartman o add android to the build
2012-02-12 23:19 Greg Kroah-Hartman o LTTng patches added
2012-02-12 23:11 Greg Kroah-Hartman o android patches
2012-02-12 23:11 Greg Kroah-Hartman o Add patch for -ltsi in kernel version
2012-02-12 23:10 Greg Kroah-Hartman o Added a script to generate android patches from mainline.
2012-02-12 22:56 Greg Kroah-Hartman I Added series and README file to start the repo off.

[main] 264a08866119c2dd919620eff67686ef59f7ace1 - commit 1 of 21 (100%)
```

Current LTSI component (as of end of May)

Extra stuff added on top of community 3.0-LTS

- patches.android
- patches.armadillo800eva
- patches.ltsi
- patches.lttng
- patches.pramfs
- patches.runtime_pm
- scripts
- Some others will be merged

LTSI 3.0 release schedule


community development : mainline
kernel 3.0 released 2011-07-21
kernel 3.2 released 2012-1-4
24 month
6 month extension expected
kernel 3.0
kernel 3.0.18
sync with LTS fixes
LTSI 3.0 2012-6-30
Staging tree open target 2012-4-1
Staging tree freeze target 2012-6-1
encourage industry to send code
merge
stabilize
18 month
new LTSI staging process
LTSI project management

- 1st LTSI will be based on LTS kernel 3.0 release
- 1st LTSI release target date is set to the end of June 2012
- 1st LTSI staging tree need to be open April and May (60 days)
- 1st LTSI need to be tested one month before release
- We hope LTS 3.0 will be maintained while LTSI 3.0 is alive.

LTSI resources

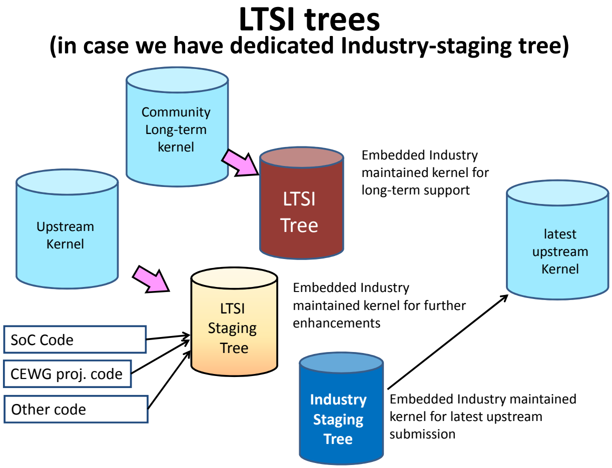
source tree (git)

- release tree = will open at the end of June
- staging tree
  http://git.linuxfoundation.org/ltsi-kernel.git
- upstreaming staging tree = will open soon

communication method

- project web = http://ltsi.linuxfoundation.org/
- mailing list
  http://lists.linuxfoundation.org/pipermail/ltsi-dev/
- social media = http://twitter.com/#!/LinuxLTSI

LTSI code repository


LTSI trees
(in case we have dedicated Industry-staging tree)
Community Long-term kernel
LTSI Tree
Embedded Industry maintained kernel for long-term support
Upstream Kernel
latest upstream Kernel
LTSI Staging Tree
Embedded Industry maintained kernel for further enhancements
SoC Code
CEWG proj. code
Other code
Industry Staging Tree
Embedded Industry maintained kernel for latest upstream submission

Conclusion

- LF/CEWG launched LTSI project to develop and distribute specially enhanced LTS kernel named LTSI for Embedded industry use. It contains some feature backport from current latest kernel, off-tree kernel patches owned by SoC vendor/product developer and others.
- We believe LTSI can dramatically reduce your own effort (=cost) for in-house kernel management. Also we hope LTSI can encourage CE company developer to send more code to upstream.
- Very first LTSI release 3.0 will be released at the end of June 2012. LTSI3.0 will be maintained (at least) until 2013 June timing.
- SCOOP!! Community maintainer decided 3.4 is the next LTS. Following this decision we will start preparation for next LTSI 3.4 staging process sometime soon.

Call for action for LTSI3.0 (now) & LTSI3.4 (next)

For SoC vendor, CPU core provider

- Send your not-yet-mainlined (AKA vendor tree) code to LTSI
- Test LTSI kernel on your environment and feedback test result
- If any software workaround exist, please share that with us.

For product producer

- Adopt LTSI kernel to reduce your in-house maintenance cost
- If you have not-yet-mainlined bug-fix, send it to LTSI

For software distributor, integrator

- Adopt and support LTSI as your base kernel.
- Send us your feedback to improve LTSI and future upstream