Hisao Munakata

LTSI Project @ Linux Foundation, CE working group

October 5th 2015
Who am I?

- From embedded SoC provider company Renesas
- Responsible for OSS software development and delivery for R-Car series SoC
- Working with W/W car OEM and 1st tear IVI customers

- 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

---

\(^1\)CE = consumer electronics
\(^2\)LTSI = Long Term Support Initiative
What is LTS & LTSI kernel?
community longterm kernel
## Schedule based periodic kernel release record

<table>
<thead>
<tr>
<th>version</th>
<th>release date</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3.7</td>
<td>2012-12-10</td>
<td>71 days</td>
</tr>
<tr>
<td>v3.8</td>
<td>2012-02-18</td>
<td>70 days</td>
</tr>
<tr>
<td>v3.9</td>
<td>2013-04-28</td>
<td>69 days</td>
</tr>
<tr>
<td>v3.10</td>
<td>2013-06-30</td>
<td>63 days</td>
</tr>
<tr>
<td>v3.11</td>
<td>2013-09-02</td>
<td>64 days</td>
</tr>
<tr>
<td>v3.12</td>
<td>2013-11-15</td>
<td>74 days</td>
</tr>
<tr>
<td>v3.13</td>
<td>2014-01-21</td>
<td>67 days</td>
</tr>
<tr>
<td>v3.14</td>
<td>2014-03-30</td>
<td>68 days</td>
</tr>
<tr>
<td>v3.15</td>
<td>2014-06-08</td>
<td>70 days</td>
</tr>
<tr>
<td>v3.16</td>
<td>2014-08-03</td>
<td>54 days</td>
</tr>
<tr>
<td>v3.17</td>
<td>2014-10-05</td>
<td>63 days</td>
</tr>
<tr>
<td>v3.18</td>
<td>2014-12-07</td>
<td>63 days</td>
</tr>
<tr>
<td>v3.19</td>
<td>2015-02-08</td>
<td>62 days</td>
</tr>
<tr>
<td>v4.0</td>
<td>2015-04-12</td>
<td>63 days</td>
</tr>
<tr>
<td>v4.1</td>
<td>2015-06-21</td>
<td>70 days</td>
</tr>
<tr>
<td>v4.2</td>
<td>2015-09-01</td>
<td>72 days</td>
</tr>
</tbody>
</table>

- **min = 54 days**
- **max = 72 days**
- **mean = 70.9 days**
You can find 1) latest released, 2) under development (=mainline, next), and several stable kernels.
upstream kernel maintenance (Stable and Longterm)

Stable

After each mainline kernel is released, it is considered "stable." Any bug fixes for a stable kernel are backported from the mainline tree and applied by a designated stable kernel maintainer. There are usually only a few bugfix kernel releases until next mainline kernel becomes available -- unless it is designated a "longterm maintenance kernel." Stable kernel updates are released on as-needed basis, usually 2-3 a month.

Longterm

There are usually several "longterm maintenance" kernel releases provided for the purposes of backporting bugfixes for older kernel trees. Only important bugfixes are applied to such kernels and they don't usually see very frequent releases, especially for older trees.

Longterm release kernels

<table>
<thead>
<tr>
<th>Version</th>
<th>Maintainer</th>
<th>Released</th>
<th>Projected EOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Greg Kroah-Hartman</td>
<td>2015-06-21</td>
<td>Sep, 2017</td>
</tr>
<tr>
<td>3.18</td>
<td>Sasha Levin</td>
<td>2014-12-07</td>
<td>Jan, 2017</td>
</tr>
<tr>
<td>3.4</td>
<td>Li Zefan</td>
<td>2012-05-20</td>
<td>Sep, 2016</td>
</tr>
<tr>
<td>3.2</td>
<td>Ben Hutchings</td>
<td>2012-01-04</td>
<td>2016</td>
</tr>
</tbody>
</table>

https://www.kernel.org/category/releases.html
Linux kernel life-cycle varies according to version

- Longterm version
  - upstream kernel migrates at around 70 days
  - longterm stable kernel selection rules
    - a new longterm kernel is picked every year.
    - a longterm kernel is maintained for 2 years and then dropped.
    - stable kernels keep the same schedule that they have been (dropping the last one after a new release happens.) These releases are best for products that require new hardware updates (desktop distros, community distros, fast-moving embedded distros (like Yocto)).
    - the normal -stable rules apply to these -longterm kernels as described in Documentation/stablekernelrules.txt

### Longterm Kernel Maintenance Last Longer Than Regular

<table>
<thead>
<tr>
<th>Version</th>
<th>Maintenance Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3.7</td>
<td>maintained till 3.7.10, then now EOL</td>
</tr>
<tr>
<td>v3.8</td>
<td>maintained till 3.8.13, then now EOL</td>
</tr>
<tr>
<td>v3.9</td>
<td>maintained till 3.9.11, then now EOL</td>
</tr>
<tr>
<td><strong>√</strong> v3.10</td>
<td>longterm stable (3.10.88), kept maintained</td>
</tr>
<tr>
<td>v3.11</td>
<td>maintained till 3.11.10, then now EOL</td>
</tr>
<tr>
<td>v3.12</td>
<td>longterm stable (3.12.47), kept maintained (by SUSE)</td>
</tr>
<tr>
<td>v3.13</td>
<td>maintained till 3.13.11 then now EOL</td>
</tr>
<tr>
<td><strong>√</strong> v3.14</td>
<td>longterm stable (3.14.52), kept maintained</td>
</tr>
<tr>
<td>v3.15</td>
<td>maintained till 3.15.10, then now EOL</td>
</tr>
<tr>
<td>v3.16</td>
<td>maintained till 3.16.7, then now EOL</td>
</tr>
<tr>
<td>v3.17</td>
<td>maintained till 3.17.8, then now EOL</td>
</tr>
<tr>
<td>v3.18</td>
<td>longterm stable (3.18.21), kept maintained (by Debian)</td>
</tr>
<tr>
<td>v3.19</td>
<td>maintained till 3.19.8, then now EOL</td>
</tr>
<tr>
<td>v4.0</td>
<td>stable release (4.1.9), till 4.2 release</td>
</tr>
<tr>
<td><strong>√</strong> v4.1</td>
<td>next longterm stable version (4.1.7)</td>
</tr>
</tbody>
</table>
Stable release include **MUST APPLY** essential fixes

<table>
<thead>
<tr>
<th>version</th>
<th>fixes</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3.7 -&gt; v3.7.10</td>
<td>718</td>
</tr>
<tr>
<td>v3.8 -&gt; v3.8.13</td>
<td>996</td>
</tr>
<tr>
<td>v3.9 -&gt; v3.9.11</td>
<td>746</td>
</tr>
<tr>
<td>v3.10 -&gt; v3.10.88</td>
<td>4,849</td>
</tr>
<tr>
<td>v3.11 -&gt; v3.11.10</td>
<td>677</td>
</tr>
<tr>
<td>v3.12 -&gt; v3.12.47</td>
<td>5,235</td>
</tr>
<tr>
<td>v3.13 -&gt; v3.13.11</td>
<td>903</td>
</tr>
<tr>
<td>v3.14 -&gt; v3.14.52</td>
<td>3,765</td>
</tr>
<tr>
<td>v3.15 -&gt; v3.15.10.</td>
<td>703</td>
</tr>
<tr>
<td>v3.16 -&gt; v3.16.7.</td>
<td>871</td>
</tr>
<tr>
<td>v3.17 -&gt; v3.17.8.</td>
<td>884</td>
</tr>
<tr>
<td>v3.18 -&gt; v3.18.21</td>
<td>2,114</td>
</tr>
<tr>
<td>v3.19 -&gt; v3.19.8.</td>
<td>873</td>
</tr>
<tr>
<td>v4.0 -&gt; v4.0.9.</td>
<td>757</td>
</tr>
<tr>
<td>v4.1 -&gt; v4.1.7.</td>
<td>697</td>
</tr>
</tbody>
</table>

- 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_BROKEN), an oops,
  a hang, data corruption, a real security issue, or some
  "oh, that's not good" issue. **In short, something critical.**
- Serious issues as reported by a user of a distribution
  kernel may also be considered if they fix a notable
  performance or interactivity issue. As these fixes are
  not as obvious and have a higher risk of a subtle
  regression they should only be submitted by a distribu-
  tion kernel maintainer and include an addendum
  linking to a bugzilla entry if it exists and additional
  information on the user-visible impact.
- New device IDs and quirks are also accepted.
- No "theoretical race condition" issues, unless an
  explanation of how the race can be exploited is also
  provided.
- It cannot contain any "trivial" fixes in it (spelling
  changes, whitespace cleanups, etc).
- It must follow **Documentation/SubmittingPatches** rules.
- It or an equivalent fix must already exist in Linus' tree
  (upstream).
Target kernel selection rules:

- Maintainer will **choose one LTS version per year**
- **Maintain it for 2 years** from its original release
- LTS-3.10 becomes EOL when LTS-4.1 is released
- Then, we have 2 LTS kernels versions like 3.14 and 4.1
LTSI kernel concept
Creating **common ground** for embedded industry

- no consolidation point in embedded Linux world

- periodic LTS release (one version per year)

Before LTSI start, each embedded project picked random kernel version.

- bug case information and workaround
- missing feature (error recovery)
- feature enhancement
- research and trial report
Renesas developed and delivered R-Car gen2 Linux BSP with LTSI-3.10 kernel. Here is the history of upstream 3.10 development and device release.

No chance to put the R-Car M2 device support to 3.10 upstream LTS kernel code

LTSI kernel was the only option to backport mainlined code
Seems some people confuse **LTS** and **LTSI** difference

LTS kernel selection by community

CVE database

new feature

security fix

bug fix

upstream development

security fix w/ “stable flag”

serious & completely proven bug fix

incrementally applying newly found fix patch

requested new feature backport

in-house patch (after review)

\[ \text{LTSI} = \text{LTS} + \text{brand new feature for cutting-edge silicon} \]
LTSI Status Update (3.14 and 4.1)
LTSI 3.14 is the latest release @ January 9, 2015
# LTSI 3.14 development history (result)

<table>
<thead>
<tr>
<th>Item</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel 3.14 merge window open</td>
<td>2014.1.19</td>
</tr>
<tr>
<td>Kernel 3.14 merge window close</td>
<td>2014.2.2</td>
</tr>
<tr>
<td>Kernel 3.14 release</td>
<td>2014.3.30</td>
</tr>
<tr>
<td>Announce of 2014 LTS kernel version</td>
<td>2014.5.20 (LinuxCon JP)</td>
</tr>
<tr>
<td>LTSI-3.14 git tree created</td>
<td>2014.5.20</td>
</tr>
<tr>
<td>3.14 becomes LTS (=3.16 release)</td>
<td>2014.8.3</td>
</tr>
<tr>
<td>LTSI-3.14 merge window open</td>
<td>2014.8.23</td>
</tr>
<tr>
<td>Patch collection period</td>
<td>84 days</td>
</tr>
<tr>
<td>LTSI-3.14-rc1 (=merge window close)</td>
<td>2014.11.14</td>
</tr>
<tr>
<td>Validation period</td>
<td>56 days</td>
</tr>
<tr>
<td>LTSI-3.14 release</td>
<td>2015.1.9</td>
</tr>
</tbody>
</table>
Greg.K.H announced next LTS is 4.1 and LTSI follow it.
# LTSI 4.1 development schedule *(merge close soon)*

<table>
<thead>
<tr>
<th>item</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel 4.1 merge window open</td>
<td>2015.4.12</td>
</tr>
<tr>
<td>kernel 4.1 merge window close</td>
<td>2015.4.26</td>
</tr>
<tr>
<td>kernel 4.1 release</td>
<td>2015.6.21</td>
</tr>
<tr>
<td>Greg announced 4.1 is next LTS(I)</td>
<td><strong>LinuxCon NA 2015</strong></td>
</tr>
<tr>
<td>LTSI-4.1 merge window open</td>
<td>2015.8.17</td>
</tr>
<tr>
<td>patch collection period</td>
<td><strong>76 days</strong></td>
</tr>
<tr>
<td>LTSI-4.1 merge window close (target)</td>
<td>2015.10.31</td>
</tr>
<tr>
<td>validation period</td>
<td><strong>50+ days</strong></td>
</tr>
<tr>
<td>LTSI-4.1 release (target)</td>
<td><strong>2015.12.20</strong></td>
</tr>
</tbody>
</table>

Compare to the previous, we adopted relatively fresh kernel
Getting closer to the latest community kernel
## Patch submission for LTSI-4.1 (as of today)

**Altera SoCFPGA (4.2 -> 4.1.6 backport)**
- Support for Arria10 platform
- Support of EDAC driver
- Support for Suspend-to-RAM

**Toshiba (planing)**
- flush file system related

**Renesas SoCs (4.2 -> 4.1.6 backport)**
- ak4642
- cpg
- gpio-rcar
- i2c-rcar
- i2c-sh-mobile
- irq-renesas-intc-irqpin
- irq-renesas-irqc
- mach-shmobile
- phy-rcar-gen2
- phy-rcar-gen2-usb
- pinctrl/sh-pfc
- rcar-dmac
- rcar-du/snd/vin
BSP development w/LTSI kernel
Renesas R-Car generation2 SoC case
R-Car gen2 Linux BSP **whole development process**

1. **Renesas upstream development team** continuous code submission to the community ML

2. LTSI development (patch backport to 3.10) 286 patches

3. LTSI-3.10
   - We adopt LTSI 3.10 kernel as a baseline for R-Car gen2 BSP

4. Reaesas backport public git
   - BSP update information
     - new fixes
     - enhance

5. R-Car BSP early drop
   - R-Car BSP v.0.8
   - R-Car BSP v.0.9
   - R-Car BSP v.1.0

---

Hisao Munakata

LTSI project update (2015 fall editoin)
Initially, Renesas adopts **upstream first strategy**

<table>
<thead>
<tr>
<th>pros</th>
<th>cons</th>
</tr>
</thead>
</table>
| - clean code (reviewed by the community)  
- coordinated with existing code  
- merged to the upstream code  
- no need to keep in-house code | - take time (roughly 6 month)  
- might need iterative approach (bit by bit)  
- might need code adjustment |

Initially, Renesas adopts upstream first strategy

- clean code (reviewed by the community)  
- coordinated with existing code  
- merged to the upstream code  
- no need to keep in-house code

Renesas adopts upstream first strategy

- clean code (reviewed by the community)  
- coordinated with existing code  
- merged to the upstream code  
- no need to keep in-house code

Initially, Renesas adopts upstream first strategy

- clean code (reviewed by the community)  
- coordinated with existing code  
- merged to the upstream code  
- no need to keep in-house code

Initially, Renesas adopts upstream first strategy

- clean code (reviewed by the community)  
- coordinated with existing code  
- merged to the upstream code  
- no need to keep in-house code
(1) Upstream development (contentious effort)

1. Renesas upstream development team continuing code submission to the community ML.

2. LTSI development (patch backport to 3.10) with 286 patches.

3. We adopt LTSI 3.10 kernel as a baseline for R-Car gen2 BSP.

4. BSP update information:
   - new fixes
   - enhance

5. LTSI-3.4 based early release.
(2) Pick a **LTSI-3.10 kernel** as a baseline

1. Renesas upstream development team continuous code submission to the community ML
2. LTSI development (patch backport to 3.10)
3. We adopt LTSI 3.10 kernel as a baseline for R-Car gen2 BSP
4. BSP update information
   - new fixes
   - enhance
5. LTSI-3.4 based early release

Renesas R-Car generation2 SoC case
Lessons learned form the real life
LTSI enhance
(3) continuous backport after LTSI merge closed

1. Renesas upstream development team continuous code submission to the community ML

2. LTSI development (patch backport to 3.10)

3. We adopt LTSI 3.10 kernel as a baseline for R-Car gen2 BSP

4. BSP update information
   - new fixes
   - enhance

5. LTSI-3.4 based early release

Renesas R-Car generation2 SoC case
Lessons learned form the real life
LTSI enhance

Hisao Munakata
LTSI project update (2015 fall editoin)
The LTS & LTSI kernel refers to a system for maintaining a long-term supported kernel. BSP development with LTSI kernel is an important aspect of this system, as it involves developing bootloader and device drivers for specific SoC (System on Chip) platforms. The conclusion and resources section highlights the value of LTSI in enhancing the development process.

Renesas R-Car generation 2 SoC case study demonstrates the practical application of LTSI in real-life scenarios. Lessons learned from this case include the importance of continuous backporting after LTSI merge. The provided Git query shows a commit for reverting an incorrect calculation of second and third plane offsets when cropping parameter.

https://git.kernel.org/cgit/linux/kernel/git/horns/renesas-backport.git/log/?h=bsp/v3.10.31-ltsi/rcar-gen2-1.9.2
(4) super-long term security fix adoption

1. Renesas upstream development team continuous code submission to the community ML

2. LTSI development (patch backport to 3.10)
   - 286 patches

3. LTSI-3.10
   - We adopt LTSI 3.10 kernel as a baseline for R-Car gen2 BSP

4. Renesas backport public git
   - BSP update information
     - new fixes
     - enhance

5. LTSI-3.4 based early release
   - R-Car BSP early drop
   - R-Car BSP v.0.8
   - R-Car BSP v.0.9
   - R-Car BSP v.1.0
We continue check if new fixes is available

- automated upstream patch scan from git
- crawling scope is own code or modified code
- F : patch severity parsing
  - H : crash, dead, freeze, hang, hung, leak, oops, panic
  - M : err, error, fix, fixes, fixed, bug
  - L : tidyup, typo, warn, warning
- S : check if this patch is cc’d to stable@vger.kernel.org
- L : check if this patch is already a part of LTSI kernel
- B : check if this patch is send to renesas-backport git
What is LTS & LTSI kernel?
BSP development w/LTSI kernel
Conclusion and Resources

BSP maintenance: **new bug-fix patch tracking**

actual scan result example (comparing upstream 3.10..3.19)

---

**LTSI stat**

Kernel Version: v3.10..v3.19
update: 2015/03/10

F H (crach dead freeze hang hung leak oops panic) M (err error fix fixes fixed bug) L (tidyup typo warn warning)
S posted to stable@vger.kernel.org ML
L LTSI 3.10.31
B backport/v3.10.31-ltsi/snapshot20

**DMAEngine**

<table>
<thead>
<tr>
<th>F</th>
<th>S</th>
<th>L</th>
<th>B</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: shdma: fix a race condition in __ld_cleanup()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: Remove_owner field for driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: sh: Remove chancnt affectations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dma: sh: drop owner assignment from platform_drivers</td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td>dma: rcar-audmapp: Fix for no corresponding slave ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: Remove the context argument to the prep_dma_cyclic_operation</td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: shdma: Allocate cyclic sg list dynamically</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: shdma: Make channel filter ignore unrelated devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: sh: Rework Kconfig and Makefile</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dmaengine: shdma: Use defines instead of hardcoded numbers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dma: rcar-audmapp: add DT support</td>
</tr>
</tbody>
</table>
(5) local in-house patch adoption and elimination

1. Renesas upstream development team continuous code submission to the community ML.
2. LTSI development (patch backport to 3.10).
3. We adopt LTSI 3.10 kernel as a baseline for R-Car gen2 BSP.
4. BSP update information:
   - new fixes
   - enhance
5. LTSI-3.10 public git

- R-Car BSP early drop
- 286 patches
- LTSI-3.4 based early release
- R-Car BSP v.0.8
- R-Car BSP v.0.9
- R-Car BSP v.1.0

Hisao Munakata

LTSI project update (2015 fall edition)
we try to eliminate in-house code from our BSP, however

- Due to time constraint (=up to six months needed for upstream-first attempt), we still need to manage some in-house patch.

- We start up-porting challenge (in-house code to the upstream flow) to eliminate (at least reduce) in-house code.

- It requires an extra code polish to comply with latest mainline kernel patch adoption criteria. But we believe this is the valuable challenges.
Lessons learned form the real life
We noticed each customer modified kernel a lot

Renesas did deliver single R-Car BSP to the all customer projects, that utilize LTSI-3.10 kernel and later code. However...

**LTSI 3.10 based BSP**

Each customer needed to modify to satisfy their product demands.

Then, again it caused fragmentation

**LTSI-3.10 kernel was not ready to develop product?**
upstream kernel is **not designed for production?**

- Driver code must be written portable and unified
- And it can not be tie with specific hw implementation
  - assumption of specific hw feature
  - use-case specific error recovery algorithm
  - allocation of dedicated acceleration engine like DMA

- Security fix done by the community is not good enough?

**Maybe we may need distribution like enterprise world**
The production kernel needs **landing procedure**

**Distribution** = verified collection of various Linux programs (=packages)

- per-build binary distribution = Debian, Ubuntu, Cent, Fedora
- source code distribution = Gentoo, Open Embedded (yocto),...

**SDK (Software Development Kit)** = subset of distribution designed for specific application = Android, Tizen IVI, MPD, Drone Code

**BSP (Board Support Package)** = subset of SDK, designed for specific target hardware. **Embedded SoC vendor develops BSP for their reference platform. Product producer modify BSP to fit with product hw**
LTSI enhance
What should be added / **dropped** for future LTSI?

**LTSI kernel**
- backport code
- new device
- new feature
- common code
  - security fix
  - patch adoption

**upstream longterm kernel**
- ?
  - drop some upstream code for certification (safety?)
Conclusion and Resources
Conclusion
We have LTSI-3.10 and 3.14 for now and start patch collection for next LTSI-4.1 release.

Version gap between LTS(I) and upstream becomes narrow at 4.1 development. This is a good thing, but...

We adopted LTSI-3.10 kernel for the production BSP, and noticed many customers needed to enhance LTSI kernel for various purpose. We want to discuss how can we improve the usability of LTSI kernel.
Resources = ltsi.linuxfoundation.org

- LTSI process document (new) =

- 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

- twitter = @LinuxLTSI

- document archives = http://ltsi.linuxfoundation.org/resources