The kernel’s limits to growth

(Linaro Connect 2017)

Jonathan Corbet
LWN.net
corbet@lwn.net
Quite frankly, this particular discussion (and others before it) has just made me irritable, and is ADDING pressure. Instead, I'd suggest that if you have a complaint about how I handle patches, you think about what I end up having to deal with for five minutes.

Go away, people. Or at least don't Cc me any more. I'm not interested, I'm taking a vacation, and I don't want to hear about it any more. In short, get the hell out of my mailbox.

— Linus Torvalds
Quite frankly, this particular discussion (and others before it) has just made me irritable, and is ADDING pressure. Instead, I'd suggest that if you have a complaint about how I handle patches, you think about what I end up having to deal with for five minutes.

Go away, people. Or at least don't Cc me any more. I'm not interested, I'm taking a vacation, and I don't want to hear about it any more. In short, get the hell out of my mailbox.

— Linus Torvalds, September 1998
Development process scalability
More recently

2.2.0: 1999-01-16
2.4.0: 2001-01-04
2.6.0: 2003-12-17
More recently

2.2.0: 1999-01-16
2.4.0: 2001-01-04
2.6.0: 2003-12-17

The fun of those days
  Massive backporting of 2.6 patches to 2.4
  Vendor Frankenstein kernels
  Lots of out-of-tree code shipped
  Painful upgrades
So what did we do?

The “upstream first” rule
So what did we do?

The “upstream first” rule

Distributed source-code control
So what did we do?

The “upstream first” rule

Distributed source-code control
  (...actually, *any* source-code control...)
So what did we do?

The “upstream first” rule

Distributed source-code control
  (...actually, any source-code control...)

The “new” release model
So what did those changes do for us?
## Recent releases

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Days</th>
<th>Devs</th>
<th>Changesets</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Mar 13</td>
<td>63</td>
<td>1,537</td>
<td>12,080</td>
</tr>
<tr>
<td>4.6</td>
<td>May 15</td>
<td>63</td>
<td>1,678</td>
<td>13,517</td>
</tr>
<tr>
<td>4.7</td>
<td>Jul 17</td>
<td>70</td>
<td>1,582</td>
<td>12,283</td>
</tr>
<tr>
<td>4.8</td>
<td>Oct 2</td>
<td>70</td>
<td>1,597</td>
<td>13,382</td>
</tr>
<tr>
<td>4.9</td>
<td>Dec 11</td>
<td>70</td>
<td>1,729</td>
<td>16,216</td>
</tr>
<tr>
<td>4.10</td>
<td>Feb 19</td>
<td>70</td>
<td>1,672</td>
<td>13,029</td>
</tr>
</tbody>
</table>
# Recent releases

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Days</th>
<th>Devs</th>
<th>Changesets</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Mar 13</td>
<td>63</td>
<td>1,537</td>
<td>12,080</td>
</tr>
<tr>
<td>4.6</td>
<td>May 15</td>
<td>63</td>
<td>1,678</td>
<td>13,517</td>
</tr>
<tr>
<td>4.7</td>
<td>Jul 17</td>
<td>70</td>
<td>1,582</td>
<td>12,283</td>
</tr>
<tr>
<td>4.8</td>
<td>Oct 2</td>
<td>70</td>
<td>1,597</td>
<td>13,382</td>
</tr>
<tr>
<td>4.9</td>
<td>Dec 11</td>
<td>70</td>
<td>1,729</td>
<td>16,216</td>
</tr>
<tr>
<td>4.10</td>
<td>Feb 19</td>
<td>70</td>
<td>1,672</td>
<td>13,029</td>
</tr>
</tbody>
</table>

Since 4.5: 79,000 changesets from 4,100 devs
Developers contributing to each release

- Total
- First-time
The Linux kernel is everywhere
We would appear to be on a roll...
We would appear to be on a roll…

So why am I worried?
“Roads and bridges”

Nadia Eghbal

We are not paying sufficient attention to the needs of our maintainers
Unpaid maintenance?

Maint. support v4.5..

20% Red Hat
11% Linux Foundation
10% Intel
  8% Linaro
  8% Google
  4% Samsung
  4% —
  3% IBM
  2% SUSE
<table>
<thead>
<tr>
<th>Maint. support v4.5..</th>
<th>Core maint support</th>
</tr>
</thead>
<tbody>
<tr>
<td>20% Red Hat</td>
<td>30% Google</td>
</tr>
<tr>
<td>11% Linux Foundation</td>
<td>21% Red Hat</td>
</tr>
<tr>
<td>10% Intel</td>
<td>9% SUSE</td>
</tr>
<tr>
<td>8% Linaro</td>
<td>7% consultants</td>
</tr>
<tr>
<td>8% Google</td>
<td>6% Facebook</td>
</tr>
<tr>
<td>4% Samsung</td>
<td>6% Intel</td>
</tr>
<tr>
<td>4% —</td>
<td>4% Huawei</td>
</tr>
<tr>
<td>3% IBM</td>
<td>3% Linutronix</td>
</tr>
<tr>
<td>2% SUSE</td>
<td>(0.7% Linaro)</td>
</tr>
</tbody>
</table>
Work nobody will pay for

Much core-kernel work
Documentation
Configuration system
Debloating
Security
...

Security worries

We have no “security officer”
no security training
no security documentation
The year in CVE numbers

CVE-2016-0723 CVE-2016-0728 CVE-2016-0758 CVE-2016-0774 CVE-2016-0821 CVE-2016-0823
CVE-2016-1237 CVE-2016-1575 CVE-2016-1576 CVE-2016-1583 CVE-2016-2053 CVE-2016-2059
CVE-2016-2061 CVE-2016-2062 CVE-2016-2063 CVE-2016-2064 CVE-2016-2065 CVE-2016-2066
CVE-2016-2067 CVE-2016-2068 CVE-2016-2069 CVE-2016-2070 CVE-2016-2085 CVE-2016-2117
CVE-2016-2143 CVE-2016-2184 CVE-2016-2185 CVE-2016-2186 CVE-2016-2187 CVE-2016-2188
CVE-2016-2383 CVE-2016-2384 CVE-2016-2543 CVE-2016-2544 CVE-2016-2545 CVE-2016-2546
CVE-2016-2547 CVE-2016-2548 CVE-2016-2549 CVE-2016-2550 CVE-2016-2782 CVE-2016-2847
CVE-2016-2853 CVE-2016-2854 CVE-2016-3070 CVE-2016-3134 CVE-2016-3135 CVE-2016-3136
CVE-2016-3137 CVE-2016-3138 CVE-2016-3139 CVE-2016-3140 CVE-2016-3156 CVE-2016-3157
CVE-2016-3672 CVE-2016-3689 CVE-2016-3707 CVE-2016-3713 CVE-2016-3841 CVE-2016-3951
CVE-2016-3955 CVE-2016-3961 CVE-2016-4440 CVE-2016-4470 CVE-2016-4482 CVE-2016-4485
CVE-2016-4486 CVE-2016-4557 CVE-2016-4558 CVE-2016-4565 CVE-2016-4568 CVE-2016-4569
CVE-2016-4578 CVE-2016-4580 CVE-2016-4581 CVE-2016-4794 CVE-2016-4805 CVE-2016-4913
CVE-2016-4951 CVE-2016-4997 CVE-2016-4998 CVE-2016-5243 CVE-2016-5244 CVE-2016-5340
CVE-2016-5342 CVE-2016-5344 CVE-2016-5400 CVE-2016-5412 CVE-2016-5696 CVE-2016-5728
CVE-2016-5828 CVE-2016-5829 CVE-2016-6130 CVE-2016-6136 CVE-2016-6156 CVE-2016-6162
CVE-2016-6187 CVE-2016-6197 CVE-2016-6198 CVE-2016-6480 [...]

 [...]
int tcp_rcv_state_process(struct sock *sk, struct sk_buff *skb, ...
    * But, this leaves one open to an easy denial of ...
    * service attack, and SYN cookies can't defend ...
    * against this problem. So, we drop the data ...
    * in the interest of security over speed. ...
    +     * in the interest of security over speed unless ...
    +     * it's still in use. ...
    */
-    goto discard;
+    kfree_skb(skb);
+    return 0;
diff --git a/net/ipv4/tcp_input.c b/net/ipv4/tcp_input.c
index c701f6a..5c16e24 100644
--- a/net/ipv4/tcp_input.c
+++ b/net/ipv4/tcp_input.c
@@ -4420,9 +4420,11 @@ int tcp_rcv_state_process(struct sock *sk, struct sk_buff *skb,
                   * But, this leaves one open to an easy denial of
                   * service attack, and SYN cookies can't defend
                   * against this problem. So, we drop the data
-                   * in the interest of security over speed.
+                   * in the interest of security over speed unless
+                   * it's still in use.
                   */
-                    goto discard;
+                    kfree_skb(skb);
+                    return 0;

}

diff --git a/net/dccp/input.c b/net/dccp/input.c
index ba34718..8fecd2d 100644
--- a/net/dccp/input.c
+++ b/net/dccp/input.c
@@ -606,7 +606,8 @@ int dccp_rcv_state_process(struct sock *sk, struct sk_buff *skb,
+                         if (inet_csk(sk)->icsk_af_ops->conn_request(sk,
+                                             skb) < 0)
+                               return 1;
+                           goto discard;
+                           consume_skb(skb);
+                           return 0;
+                       }
+                   if (dh->dccph_type == DCCP_PKT_RESET)
+                       goto discard;
tcp_rcv_state_process(struct sock *sk, struct sk_buff *skb,
* But, this leaves one open to an easy denial of
* service attack, and SYN cookies can't defend
* against this problem. So, we drop the data
* in the interest of security over speed.
* in the interest of security over speed unless
* it's still in use.
* /
- goto discard;
+ consume_skb(skb);
+ return 0;
}

if (dh->dccph_type == DCCP_PKT_RESET)
goto discard;
WHERE'S MY CAR? —

Researchers discover security problems under the hood of automobile apps

Kaspersky researchers find Android apps for connected cars soft targets for hackers.

SEAN GALLAGHER - FEB 17, 2017 6:29 PM UTC
Security shows a big hole in our maintainer model
What is maintainership?
How does one become a maintainer?

Maintainers tend to get to be maintainers because they were good at something else, and not good enough at hiding from the "maintainer" role. There is a paradox here as a maintainer must be good at saying "No", but if they were they might never have agreed to become a maintainer.

— Neil Brown
How does one stop?

I’m trying to appear to be an incompetent maintainer so that someone will offer to take over. It isn’t working yet.

— Neil Brown
How does one stop?

I’m trying to appear to be an incompetent maintainer so that someone will offer to take over. It isn’t working yet.
— Neil Brown

I have decided to fall back on the mechanism by which I ended up being maintainer in the first place. I will create a vacuum and hope somebody fills it.
— Neil Brown
What is a maintainer’s authority?

You should always be able to handle other people changing files in your area at any point in time. Kernel maintainership is not “no one else can ever touch this!” type of development. — Greg Kroah-Hartman
What is a maintainer’s authority?

You should always be able to handle other people changing files in your area at any point in time. Kernel maintainership is not “no one else can ever touch this!” type of development. — Greg Kroah-Hartman

It is *my* prerogative to say no to anything in arch/arm, and I really don’t have to give reasons for it if I choose to. — Russell King
“A bunch of little fiefdoms”
What are a maintainer’s responsibilities?

I can’t take patches without a changelog text, and neither should any other maintainer.
— Greg Kroah-Hartman
What are a maintainer’s responsibilities?

I can’t take patches without a changelog text, and neither should any other maintainer.
— Greg Kroah-Hartman

(536 no-changelog patches were merged for 4.10)
What are a maintainer’s responsibilities?

Review the code
Mentor developers
Respond quickly to patches
Check code provenance
Respond to regressions
Route fixes to -stable
Represent the subsystem to the world
Resist company pressure
Keep Linus happy
[...]
Patch management

Not dropping patches through the cracks
Proper Git repository practices
Informing contributors about actions
Avoiding / handling conflicts
...

...
Speaking of patch management

Kids these days do things differently.

Photo: Lars Plougmann
Our maintainers are getting older
Back to the point

We don’t define the maintainer role well
We don’t document how to fill it
We don’t train future maintainers
Back to the point

We don’t define the maintainer role well
We don’t document how to fill it
We don’t train future maintainers

How much more can we scale in this mode?
Some other concerns
Review bandwidth

The big problem is this, we really only have a very small group of people reviewing code in the kernel community.

— Greg Kroah-Hartman
The big problem is this, we really only have a very small group of people reviewing code in the kernel community.
— Greg Kroah-Hartman, 2006
The big problem is this, we really only have a very small group of people reviewing code in the kernel community.
— Greg Kroah-Hartman, 2006

I am worried that the number of patches posted to linux-mm grows over time while the number of reviewers doesn’t scale up with that trend.
— Michal Hocko, 2017
Wolfram Sang: the number of reviewers is not scaling with the number of contributors.
As a consequence

Maintainers burn out and fall behind
As a consequence

Maintainers burn out and fall behind

Unreviewed code gets in
I’m seriously grumpy about this engineering trainwreck, which has seven SOBs from [$COMPANY] developers for 50 lines of code. And none of them figured out that this is broken. Impressive fail!
— Thomas Gleixner
As a consequence

Maintainers burn out and fall behind

Unreviewed code gets in

Long-term API problems
Review bandwidth is a problem for all projects
We work hard to encourage contributions

Perhaps we should do more to promote code-review skills?
Out-of-tree code
# Mobile SoC code out-of-tree

<table>
<thead>
<tr>
<th>Company</th>
<th>Phone</th>
<th>SOC</th>
<th>Insertions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG</td>
<td>G3</td>
<td>Msm</td>
<td>2.6 M</td>
</tr>
<tr>
<td>Motorola</td>
<td>Moto X</td>
<td>Msm</td>
<td>1.8 M</td>
</tr>
<tr>
<td>Samsung</td>
<td>Galaxy 4</td>
<td>Exynos</td>
<td>1.1 M</td>
</tr>
<tr>
<td>Samsung</td>
<td>Galaxy S5</td>
<td>Msm</td>
<td>3.1 M</td>
</tr>
<tr>
<td>Sony</td>
<td>Xperia Z2</td>
<td>Msm</td>
<td>1.8 M</td>
</tr>
<tr>
<td>Sony</td>
<td>Xperia C</td>
<td>Mediatek</td>
<td>1.9 M</td>
</tr>
<tr>
<td>Acer</td>
<td>Liquid E2</td>
<td>Mediatek</td>
<td>1.4 M</td>
</tr>
<tr>
<td>Asus</td>
<td>Zenfone 6</td>
<td>Atom</td>
<td>2.2 M</td>
</tr>
<tr>
<td>Huawei</td>
<td>Ascend P7</td>
<td>Hisilicon</td>
<td>2.7 M</td>
</tr>
</tbody>
</table>
Out-of-tree code consequences

Bugs and security issues
Inability to run mainline kernels
Maintainer stress
Maintainers pulled out of the community
More recently

2.2.0: 1999-01-16
2.4.0: 2001-01-04
2.6.0: 2003-12-17

The fun of those days
  Massive backporting of 2.6 patches to 2.4
  Vendor Frankenstein kernels
  Lots of out-of-tree code shipped
  Painful upgrades
My fancy new phone

Android version
7.1.1

Kernel version
3.10.73-gfe160e5
android-build@whpo7bat.corp.google.com #1
Wed Dec 7 20:26:32 UTC 2016

Build number
N4F260
How much more can we grow with this much energy being directed away from our community?
Complexity
So what can we do?
Recognize maintainership as an activity needing support
Document what it means to be a maintainer
Create training and mentoring for new maintainers
Move away from the single-maintainer model (explore group maintainership)
Teach code-review skills and encourage their use
Pay more attention to our unmaintained dark corners
Think about our next generation of tools
Don’t assume our process-scalability problems are behind us
Thank you