Rust and Linux

Greg Kroah-Hartman

gregkh@linuxfoundation.org

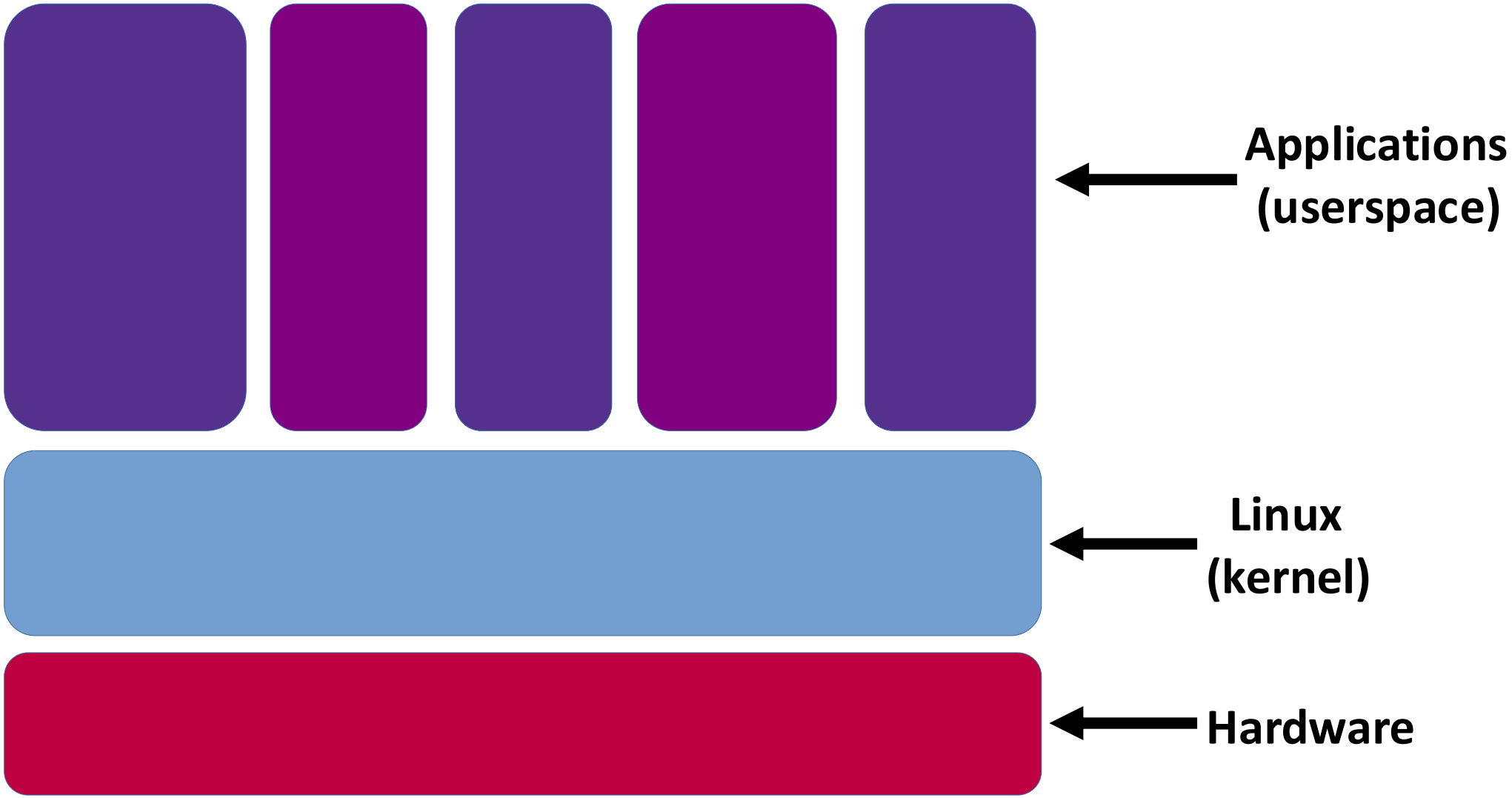
git.sr.ht/~gregkh/presentation-rust

C and Rust and Linux

Greg Kroah-Hartman

gregkh@linuxfoundation.org

git.sr.ht/~gregkh/presentation-rust


Applications
(userspace)
Linux
(kernel)
Hardware

Linux Developer Community

5,221 developers

375+ employers

2025

Linux Developer Community

78,209 changes accepted

8.9 changes per hour

40 changes per day in stable trees

13 CVEs assigned per day

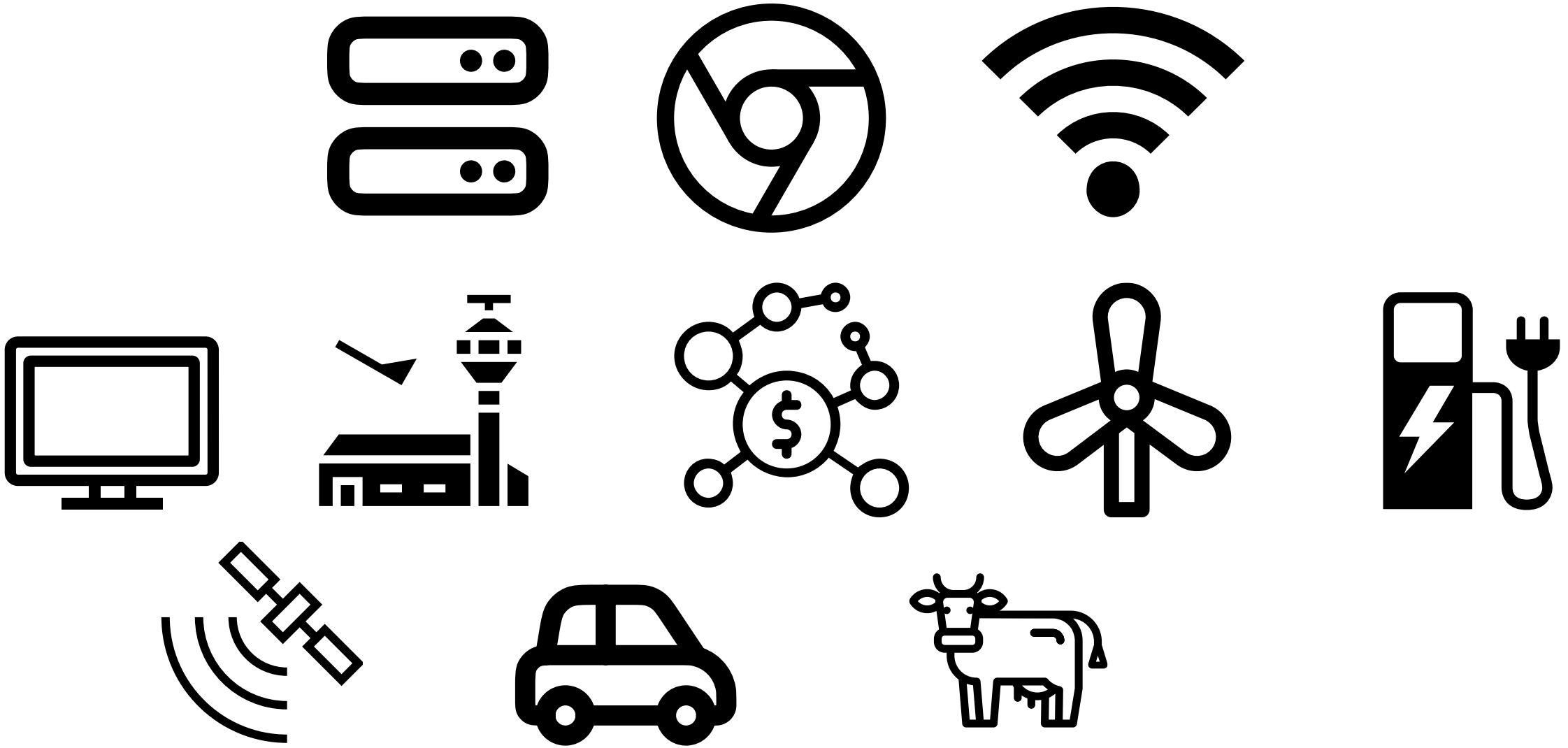
New release every 8-9 weeks

How we do it

90 minute talk, with summary:

search “pragmatic engineer Linux”

Linux runs the world



Some icons Attribution CC License, svgrepo.com

```
p = hci_conn_params_add(hdev, &cp->addr.bdaddr, addr_type);

if (p->auto_connect == HCI_AUTO_CONN_EXPLICIT)
        p->auto_connect = HCI_AUTO_CONN_DISABLED;

conn = hci_connect_le_scan(hdev, &cp->addr.bdaddr, addr_type,
                           sec_level, HCI_LE_CONN_TIMEOUT,
                           CONN_REASON_PAIR_DEVICE);
```

```
p = hci_conn_params_add(hdev, &cp->addr.bdaddr, addr_type);
if (!p) {
        err = -EIO;
        goto unlock;
}

if (p->auto_connect == HCI_AUTO_CONN_EXPLICIT)
        p->auto_connect = HCI_AUTO_CONN_DISABLED;

conn = hci_connect_le_scan(hdev, &cp->addr.bdaddr, addr_type,
                           sec_level, HCI_LE_CONN_TIMEOUT,
                           CONN_REASON_PAIR_DEVICE);
```

CVE-2024-43884

```
p = hci_conn_params_add(hdev, &cp->addr.bdaddr, addr_type);
if (!p) {
        err = -EIO;
        goto unlock;
}

if (p->auto_connect == HCI_AUTO_CONN_EXPLICIT)
        p->auto_connect = HCI_AUTO_CONN_DISABLED;

conn = hci_connect_le_scan(hdev, &cp->addr.bdaddr, addr_type,
                           sec_level, HCI_LE_CONN_TIMEOUT,
                           CONN_REASON_PAIR_DEVICE);
```

CVE-2024-43884

```
p = hci_conn_params_add(hdev, &cp->addr.bdaddr, addr_type);
if (!p) {
        err = -EIO;
        goto unlock;
}

if (p->auto_connect == HCI_AUTO_CONN_EXPLICIT)
        p->auto_connect = HCI_AUTO_CONN_DISABLED;

conn = hci_connect_le_scan(hdev, &cp->addr.bdaddr, addr_type,
                           sec_level, HCI_LE_CONN_TIMEOUT,
                           CONN_REASON_PAIR_DEVICE);
...
unlock:
        hci_dev_unlock(hdev);
        return err;
```

CVE-2024-53198

```
--- a/drivers/xen/xenbus/xenbus_probe.c
+++ b/drivers/xen/xenbus/xenbus_probe.c
@@ -313,7 +313,7 @@ int xenbus_dev_probe(struct device *_dev)
        if (err) {
                dev_warn(&dev->dev, "watch_otherend on %s failed.\n",
                         dev->nodename);
-               return err;
+               goto fail_remove;
        }
```

```
--- a/net/mac80211/ethtool.c
+++ b/net/mac80211/ethtool.c
@@ -19,16 +19,13 @@ static int ieee80211_set_ringparam(struct net_device *dev,
                                   struct netlink_ext_ack *extack)
 {
        struct ieee80211_local *local = wiphy_priv(dev->ieee80211_ptr->wiphy);
-       int ret;

        if (rp->rx_mini_pending != 0 || rp->rx_jumbo_pending != 0)
                return -EINVAL;

-       wiphy_lock(local->hw.wiphy);
-       ret = drv_set_ringparam(local, rp->tx_pending, rp->rx_pending);
-       wiphy_unlock(local->hw.wiphy);
+       guard(wiphy)(local->hw.wiphy);

-       return ret;
+       return drv_set_ringparam(local, rp->tx_pending, rp->rx_pending);
 }
```

```
@@ -183,25 +179,21 @@ static int __fwnode_iio_channel_get(struct iio_channel *channel,

-       channel = kzalloc(sizeof(*channel), GFP_KERNEL);
+       struct iio_channel *channel __free(kfree) =
+               kzalloc(sizeof(*channel), GFP_KERNEL);
        if (!channel)
                return ERR_PTR(-ENOMEM);

        err = __fwnode_iio_channel_get(channel, fwnode, index);
        if (err)
-               goto err_free_channel;
+               return ERR_PTR(err);

-       return channel;
-
-err_free_channel:
-       kfree(channel);
-       return ERR_PTR(err);
+       return_ptr(channel);
 }

 static struct iio_channel *
```

C scoped locks and allocations

C scoped locks and allocations

- Coding is simpler

C scoped locks and allocations

- Coding is simpler
- Reviewing is easier

C scoped locks and allocations

- Coding is simpler
- Reviewing is easier
- Less bugs!

C scoped locks and allocations

- Coding is simpler
- Reviewing is easier
- Less bugs!
- More fun!

```
p = hci_conn_params_add(hdev, &cp->addr.bdaddr, addr_type);

if (p->auto_connect == HCI_AUTO_CONN_EXPLICIT)
        p->auto_connect = HCI_AUTO_CONN_DISABLED;

conn = hci_connect_le_scan(hdev, &cp->addr.bdaddr, addr_type,
                           sec_level, HCI_LE_CONN_TIMEOUT,
                           CONN_REASON_PAIR_DEVICE);
```

```
p = hci_conn_params_add(hdev, cp.addr.bdaddr, addr_type)?;

if (p.auto_connect == HCI_AUTO_CONN_EXPLICIT) {
        p.auto_connect = HCI_AUTO_CONN_DISABLED;
}

conn = hci_connect_le_scan(hdev, cp.addr.bdaddr, addr_type,
                           sec_level, HCI_LE_CONN_TIMEOUT,
                           CONN_REASON_PAIR_DEVICE)?;
```

```
struct Inner {
    value: i32,
}

struct RustMiscDevice {
    inner: Mutex<Inner>,
}

fn get_value(&self, mut writer: UserSliceWriter) -> Result<isize> {
    let guard = self.inner.lock();
    let value = guard.value;
```

Rust can prevent a huge majority of security issues at build time, not review time.

Rust can prevent a huge majority of security issues at build time, not review time.

- Coding is simpler
- Reviewing is easier
- Less bugs!
- More fun!

developers > # maintainers

Rust can make reviewing code easier

- Enforce validation of untrusted data
- Enforce memory lifetime rules
- Enforce locking rules
- Enforce error handling
- Enforce type safety

Greg’s unscientific opinion

80% of Linux CVEs would be impossible if the code was written in Rust.

Rust can crash just as bad as C

```
let mut foo:Vec<u32> = Vec::new();

let _ = foo.push(1, GFP_KERNEL);
let _ = foo.push(2, GFP_KERNEL);
let _ = foo.push(3, GFP_KERNEL);
let _ = foo.push(4, GFP_KERNEL);
let _ = foo.push(5, GFP_KERNEL);

let crash = foo[foo.len()];
```

Rust is in Linux today

36,000,000 lines C

113,000 lines Rust

Change is hard

Rust forces C apis to be re-reviewed

Change can be good

Change can make maintaining easier

Make the compiler do the work

Rust == more fun for maintainers

Rust == more fun for maintainers

Rust == more secure Linux for users

World domination is proceeding as planned