

Large-Scale Code Restructuring and Tooling with Coccinelle

Michele MARTONE

Leibniz Supercomputing Centre (Garching bei München, Germany)

45th VI-HPS Tuning Workshop
LRZ, June 10, 2024

The Linux Kernel

- ▶ runs everywhere
- ▶ probably on your phone now
- ▶ mostly C language code

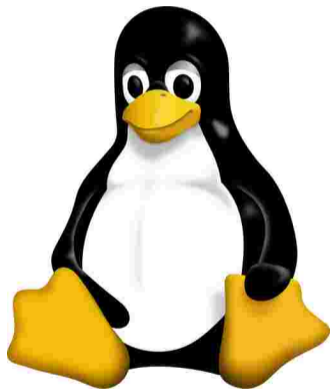


Figure: Linux mascot Tux by Larry Ewing

Bug: !x&y¹

```
1 commit e6bafba5b4765a5a252f1b8d31cbf6d2459da337
2 Author: Al Viro <viro@ZenIV.linux.org.uk>
3 Date: Wed Feb 13 04:03:25 2008 +0000
4
5     wmi: (!x & y) strikes again
6
7     Signed-off-by: Al Viro <viro@zeniv.linux.org.uk>
8     Acked-by: Carlos Corbacho <carlos@strangeworlds.co.uk>
9     Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
10
11 diff --git a/drivers/acpi/wmi.c b/drivers/acpi/wmi.c
12 index 457ed3d3f51c..efacc9f8bfe3 100644
13 --- a/drivers/acpi/wmi.c
14 +++ b/drivers/acpi/wmi.c
15 @@ -247,7 +247,7 @@ u32 method_id, const struct acpi_buffer *in, struct acpi_buffer *out)
16     block = &wblock->gblock;
17     handle = wblock->handle;
18
19 -     if (!block->flags & ACPI_WMI_METHOD)
20 +     if (!(block->flags & ACPI_WMI_METHOD))
21         return AE_BAD_DATA;
22
23     if (block->instance_count < instance)
```

¹See <https://coccinelle.gitlabpages.inria.fr/website/papers/fosdem10.pdf>

Bug: possible NULL dereference²

```
1 commit 3c8a9c63d5fd738c261bd0ceece04d9c8357ca13
2 Author: Mariusz Kozlowski <m.kozlowski@tuxland.pl>
3 Date: Sun Jul 5 19:48:35 2009 +0000
4
5     tun/tap: Fix crashes if open() /dev/net/tun and then poll() it.
6
7 ...
8 diff --git a/drivers/net/tun.c b/drivers/net/tun.c
9 index b393536012fb..027f7aba26af 100644
10 --- a/drivers/net/tun.c
11 +++ b/drivers/net/tun.c
12 @@ -486,12 +486,14 @@ static unsigned int tun_chr_poll(struct file *file, poll_table * wait)
13  {
14         struct tun_file *tfile = file->private_data;
15         struct tun_struct *tun = __tun_get(tfile);
16 -     struct sock *sk = tun->sk;
17 +     struct sock *sk;
18     unsigned int mask = 0;
19
20     if (!tun)
21         return POLLERR;
22
23 +     sk = tun->sk;
24 +
25     DBG(KERN_INFO "%s: tun_chr_poll\n", tun->dev->name);
26
27     poll_wait(file, &tun->socket.wait, wait);
```

²See <https://coccinelle.gitlabpages.inria.fr/website/papers/fosdem10.pdf>

Coccinelle, a tool *to identify* and *to eliminate* bugs



Rules with patch to fix the !x&y bug³

```
1 @@
2 expression E;
3 constant C;
4 @@
5
6 - !E & C
7 + !(E & C)
```

³See <https://coccinelle.gitlabpages.inria.fr/website/papers/fosdem10.pdf>

Rules to detect possible NULL dereference bugs (first sketch)⁴

```
1 @@
2 expression x;
3 identifier fld;
4 @@
5 * x->fld
6   . . .
7 * x == NULL
```

► *isomorphisms* cause `x == NULL` to also match e.g. `!x`.

⁴See <https://coccinelle.gitlabpages.inria.fr/website/papers/fosdem10.pdf>

Rules to detect possible NULL dereference bugs⁵

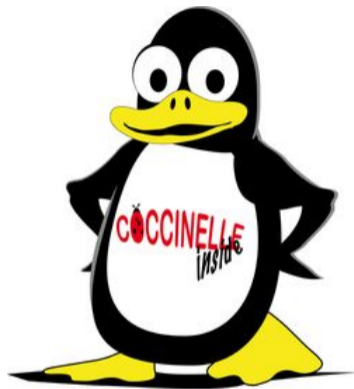
```
1 @@
2 type T;
3 identifier i, fld;
4 expression E;
5 statement S;
6 @@
7 - T i = E->fld;
8 + T i;
9     ... when != E
10         when != i
11     if (E == NULL) S
12 + i = E->fld;
```

► fewer false positives than previous rule

⁵See <https://coccinelle.gitlabpages.inria.fr/website/papers/fosdem10.pdf>

What is Coccinelle?

- ▶ regexps (sed, awk, perl, python...)?
- ▶ a *source-to-source translator*
- ▶ *semantic patching*
- ▶ safeguards Linux from bugs
- ▶ used for updating device drivers
- ▶ main author: Julia LAWALL
(Inria Paris, France)



Another use case

Suppose you have a

- ▶ large simulation code base ($> 100\text{kLoC}$) in C
- ▶ developed for many years, by many people

Lots of loop accesses

E.g.

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

What could be most obvious optimization (for CPUs)?

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

What could be most obvious optimization (for CPUs)?

Hint: `S[i]` are structures

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

Change AoS into Structure-of-Arrays (SoA)

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

⇒

```
for(i = 0; i < N; ++i)
{
    SoA.Metals[i] = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = SoA.Metals[i] * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

AoS to SoA in a Nutshell

```
1 @@
2 identifier q =~ "prsr";
3 identifier s = {P};
4 identifier i = {i,k};
5 fresh identifier s_q = s ## "_" ## q;
6 @@
7 - s[i].q
8 + s_q[i]
```

basicaos2soa1.cocci

```
0 @@ -1,9 +1,9 @@
1 struct p_t {
2     double prsr, vel;
3 };
4
5 p_t P[3];
6 int main() {
7     int i,j,k;
8 -     P[i].prsr++;
9 +     P_prsr[i]++;
10 }
```

basicaos2soa1.diff

Managing complexity on large code bases

- ▶ ease code annotation
- ▶ GPU *language extensions* (CUDA, HIP, ...)
- ▶ modern C++ features (mdspan, SYCL, ...)
- ▶ advanced `#pragma` manipulation: OpenMP, ...
- ▶ enforcing C++ coding guidelines

Ease code annotation

```
1 @@ @@
2   #include <omp.h>
3 + #include <caliper/cali.h>
4
5 @@ @@
6 + CALI_MARK_BEGIN(__func__);
7 + CALI_MARK_BEGIN("parallel");
8   #pragma omp parallel
9   {
10    ...
11  }
12 + CALI_MARK_END("parallel");
13 + CALI_MARK_END(__func__);
```

caliper1.cocci

```
0 @@ -1,8 +1,13 @@
1   #include <omp.h>
2 + #include <caliper/cali.h>
3   int f() {
4 -   #pragma omp parallel
5 +   CALI_MARK_BEGIN(__func__);
6 +   CALI_MARK_BEGIN("parallel");
7 +   #pragma omp parallel
8     {
9       // expect loops here
10    }
11 +   CALI_MARK_END("parallel");
12 +   CALI_MARK_END(__func__);
13  }
14  int main() { }
```

caliper1.diff

CUDA GPUs extensions

```
1 #spatch --smp1-spacing
2 @@
3 attribute name __device__;
4 @@
5 + __device__
6   int f(void) {
7 +   const int gthid =
8 +     blockIdx.x * blockDim.x
9 +     + threadIdx.x;
10     ...
11 }
12
13 @@
14 identifier d,b;
15 @@
16   int d;
17   int b;
18 - f();
19 + f<<<d,(d+b-1)/b>>>();
```

```
0 @@ -1,10 +1,13 @@
1 -int f(void)
2 +__device__ int f(void)
3 {
4 +   const int gthid =
5 +     blockIdx.x * blockDim.x
6 +     + threadIdx.x;
7   { /* loops */ }
8 }
9 int main(void)
10 {
11   int ds;
12   int bs;
13 - f();
14 + f<<<ds,(ds+bs-1)/bs>>>();
15 }
```

cuda.diff

C++23 multi-index operators

```
1 # spatch --c++=23
2 @tomultiindex@
3 symbol a,i,j;
4 expression k;
5 @@
6 - a[i][j][k]
7 + a[i, j, k]
8
9 @@
10 symbol b;
11 @@
12 - b[...]
13 + b[0]
```

```
0 @@ -1,8 +1,8 @@
1 int main()
2 {
3     int a[1][1][1];
4     int b[1][1][1];
5     int i=0,j=0,k=0;
6 - a[i][j][k]++;
7 - b[i][j][k]++;
8 + a[i, j, k]++;
9 + b[0][j][k]++;
10 }
```

mdspan1.diff

Decluttering with OpenMP 5.1 pragmas

```
1 @@
2 identifier c, f, l;
3 expression b, k;
4 type T;
5 @@
6
7 + const T j = k;
8 + #pragma omp unroll partial (j)
9   for (T c=0; c
10 -           + k - 1
11             < l ; ... )
12 {
13 -     f(b+0);
14 -     f(b+1);
15 -     f(b+2);
16 -     f(b+3);
17 +     f(b);
18 }
```

deunroll1.cocci

```
0 @@ -1,12 +1,11 @@
1   int f(int i){}
2   int main()
3   {
4           const int n = 13;
5 -         for (int i=0;i+4-1<n;i++)
6 +         const int j = 4;
7 +#pragma omp unroll partial (j)
8 +         for (int i=0;i<n;i++)
9           {
10 -             f(i+0);
11 -             f(i+1);
12 -             f(i+2);
13 -             f(i+3);
14 +             f(i);
15           }
16 }
```

deunroll1.diff

C++ Core Guideline F.17 (Bjarne Stroustrup, Herb Sutter)

```
1 #spatch --c++
2 @r1@
3 type T;
4 identifier f;
5 parameter list pl;
6 @@
7 T f(pl) { ... }
8
9 @r2@
10 typedef A, B;
11 type heavy_type={A,B};
12 type r1.T;
13 identifier r1.f;
14 symbol i;
15 @@
16 ///Note: heavy copy!
17 T f (
18     ..., heavy_type i, ...
19 ) { ... }
```

cppcg_f17.cocci

```
0 @@ -1,13 +1,16 @@
1 #include <array>
2 struct A { std::array<int,999> a; }; // heavy
3 struct B { std::array<int,999> a; }; // heavy
4 struct C { std::array<int, 16> a; }; // light
5 void if1(int i) {}
6 void if2(int &arg) {}
7 void if3(const int &arg) {}
8 ///Note: heavy copy!
9 void af1(A i) {}
10 ///Note: heavy copy!
11 void bf1(B i) {}
12 ///Note: heavy copy!
13 void bf2(const B i) {}
14 void bf3(const B & i) {}
15 void cf1(C i) {}
16 int main() { }
```

cppcg_f17.diff

For “in” parameters, pass cheaply-copied types by value and others by reference to const

No Raw Loops (Sean Parent)

```
1 #spatch --c++=17
2 @@ @@
3 #include <iostream>
4 + #include <algorithm>
5 + #include <functional>
6
7 @@
8 type T;
9 constant k;
10 identifier elem,result,arrid;
11 @@
12 - bool result = false;
13 ...
14 - for ( T &elem : arrid )
15 - if ( \(\ elem == k \| k == elem \) )
16 - {
17 - ...
18 - result = true;
19 - break;
20 - }
21 + const bool result =
22 + (find(begin(arrid),end(arrid),k) !=
23 + end(arrid));
```

norawloops.cocci

```
0 @@ -1,20 +1,15 @@
1 #include <vector>
2 #include <iostream>
3 + #include <algorithm>
4 + #include <functional>
5 int main()
6 {
7     using namespace std;
8     vector v = {1,2,3};
9 - bool has_zero = false;
10
11     v[2] = 0;
12
13 - for ( int &a : v )
14 - if (0 == a)
15 - {
16 -     cout << "doing things\n";
17 -
18 -     has_zero = true;
19 -     break;
20 - }
21 + const bool has_zero =
22 + (find(begin(v), end(v), 0) != v.end());
23     cout << has_zero << endl;
24 }
```

norawloops.diff

Current state

- ▶ C language covered well
- ▶ working on
 - ▶ covering C++
 - ▶ easing further use cases

Some possible uses

- ▶ writing tests / enforcing properties
- ▶ throwaway performance experiments
- ▶ clean up code
- ▶ introducing tooling

Acknowledgements

LRZ-Inria collaboration made possible by

▶ SiVeGCS Project


▶ BayFrance travel grant



Resources

 Coccinelle's GitHub page

<https://github.com/coccinelle/coccinelle>

 (also RUST version in preparation, see website)

 247-page training from 2019

https://www.lrz.de/services/compute/courses/x_lecturenotes/hspc1w19.pdf

 Misc Coccinelle things

<https://martone.userweb.mwn.de/>