Porting of drivers/md/dm-cache-policy-cleaner.c

June 5, 2017

1 Introduction

The driver drivers/md/dm-cache-policy-cleaner.c was introduced in 2013 in the commit 8735a81. It implements a cache cleanerpolicy. This driver was accompanied by changes in documentation, Kconfig, and a make file. Gcc produces 2 warnings that reduce to the same 2 warnings.

2 struct dm_cache_policy.map

Gcc reports that the field `map` of the structure `struct struct dm_cache_policy` is initialized to a value of the wrong type. We try the following patch query (step1.cocci):

```
@r1 depends on before@
identifier i,f1;
@@

struct dm_cache_policy i = {
  .map = \(f1\|&f1\),
};

@r2 depends on before expression@
struct dm_cache_policy e;
identifier f2;
@@

  <+...e.map...+> = \(f2\|&f2\)

@r3 depends on before expression@
struct dm_cache_policy *e;
identifier f3;
@@

  <+...e->map...+> = \(f3\|&f3\)

@r depends on before@
identifier f,r1.f1,r2.f2,r3.f3;
@@

(
(
 f1
|f2
|f3
```

```
)
&
 f
)

@@
type T;
identifier r.f;
@@

- T
  f(...) { ... }

@@
identifier r.f;
parameter p;
@@

f(...,
- p
  ,...) { ... }

@@
identifier r.f;
parameter p;
@@

f(...,
+ p
  ,...) { ... }
```

The only result is fb4100a at 5%. This contains two examples of `map` callback functions. A new parameter is added in the second to last position. In both of them, most of the parameters, including the new one, are passed on to another function. But there seems to be no such function call in our driver. We may conjecture that the parameter is not needed in our case, leaving the following change:

```
@r1@
identifier f1;
struct dm_cache_policy e;
@@

e.map = f1;

@@
identifier r1.f1,result;
symbol locker;
@@

f1(...,
+  struct policy_locker *locker,
   struct policy_result *result) { ... }
```

This is the only change that was made in this function in v4.6.
== success (1/1)

3 struct dm_cache_policy.writeback_work

Gcc reports that the field `writeback_work` of the structure `struct struct dm_cache_policy` is initialized to a value of the wrong type. We try the following patch query (step2.cocci):

```
@r1 depends on before@
identifier i,f1;
@@

struct dm_cache_policy i = {
  .writeback_work = \(f1\|&f1\),
};

@r2 depends on before expression@
struct dm_cache_policy e;
identifier f2;
@@

  <+...e.writeback_work...+> = \(f2\|&f2\)

@r3 depends on before expression@
struct dm_cache_policy *e;
identifier f3;
@@

  <+...e->writeback_work...+> = \(f3\|&f3\)

@r depends on before@
identifier f,r1.f1,r2.f2,r3.f3;
@@

(
(
 f1
|f2
|f3
)
&
 f
)

@@
type T;
identifier r.f;
@@

- T
  f(...) { ... }

@@
identifier r.f;
parameter p;
@@

f(...,
- p
```

```
  ,...) { ... }

@@
identifier r.f;
parameter p;
@@

f(...,
+ p
  ,...) { ... }
```

The only result is 20f6814 at 25%. Theer are only two examples of these functions. They get a new last parameter. One of the functions passes that new parameter to anoher callback function. Another does not use a callback function, nbut instead makes some direct writes. This one does not use the new parameter. Our driver also makes some direct writes. So we conjecture that using the parameter is not necessary. We thus obtain:

```
@r1@
struct dm_cache_policy e;
identifier i,f1;
@@

e.writeback_work = f1

@@
identifier r1.f1,result;
symbol locker;
@@

f1(...
+  ,bool critical_only
   ) { ... }
```

This is the only change made in the later version.
== success (1/1)