Enhancing spatial safety: fixing thousands of *-Wflex-array-member-not-at-end* warnings

Gustavo A. R. Silva
gustavo@embeddedor.com
fosstodon.org/@gustavoars
https://embeddedor.com/blog/presentations/

Supported by
The Linux Foundation & Alpha-Omega

Everything Open
January 21, 2025
Adelaide, Australia

Who am I?

By @shidokou

Who am I?

- **Upstream first** – 9 years.
- Upstream Linux Kernel Engineer.
    - Kernel hardening.
    - Proactive security.

- Kernel Self-Protection Project (**KSPP**).
- Google Open Source Security Team (**GOSST**).
    - Linux Kernel division.

By @shidokou

Agenda

- **Introduction**
  - C99 flexible-array members (FAMs)
  - The new *-Wflex-array-member-not-at-end* compiler option
- **The challenge of -Wflex-array-member-not-at-end**
  - What’s wrong with FAMs in the middle?
  - How did we get here? - A brief history
  - Fixing thousands of *-Wfamnae* warnings
  - What’s next?
- **Conclusions**

Quick review of C99 flexible-array members

– Should be the last member of a struct.

```
struct flex {
    ...
    size_t count;
    struct foo fam[];
};
```

Quick review of C99 flexible-array members

– Should be the last member of a struct.
– The flex struct usually contains a ***counter*** member.

```
struct flex {
   ...
   size_t count;
   struct foo fam[];
};
```

Quick review of C99 flexible-array members

- Should be the last member of a struct.
- The flex struct usually contains a ***counter*** member.
- ***struct*** *flex may not be a member of another struct.*

```
struct flex {
    ...
    size_t count;
    struct foo fam[];
};
```

Quick review of C99 flexible-array members

- Should be the last member of a struct.
- The flex struct usually contains a ***counter*** member.
- ***`struct`*** *`flex` may not be a member of another struct.*

```
struct flex {
    ...
    size_t count;
    struct foo fam[] __counted_by(count);
};
```

Quick review of C99 flexible-array members

- Should be the last member of a struct.
- The flex struct usually contains a ***counter*** member.
- ***struct*** *flex may not be a member of another struct.*
- Run-time bounds-checking coverage on FAMs. (link)

```
struct flex {
	...
	size_t count;
	struct foo fam[] __counted_by(count);
};
```

The new -Wflex-array-member-not-at-end

- Developed by Qing Zhao last year (2023)
- Released in GCC 14

The new -Wflex-array-member-not-at-end

– Warns about FAMs in the middle of composite structs.

```
struct flex {
    ...
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...

    struct flex middle;

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

– Warns about FAMs in the middle of composite structs.

```
struct flex {
    ...
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...

    struct flex middle;

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct flex {
    ...
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...

    struct flex middle; /* -Wfamnae warning! */

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

– Warns about FAMs in the middle of composite structs.

```
struct composite {
    ...

    struct flex last; /* This is fine! */
};
```

The new -Wflex-array-member-not-at-end

– Warns about FAMs in the middle of composite structs.

```
struct composite {
    ...

    struct flex last; /* This is fine! */
};

struct another {
    ...

    struct composite middle;

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

– Warns about FAMs in the middle of composite structs.

```
struct composite {
    ...

    struct flex last; /* This is fine! */
};

struct another {
    ...

    struct composite middle; /* -Wfamnae warning! */

    ... more objects ...
};
```

The challenge of enabling **-Wflex-array-member-not-at-end**

What’s wrong with FAMs in the middle?

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member

```
struct composite {
    ...

    struct flex last;
};
```

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member
  - **not the last member**

```
struct composite {
    ...

    struct flex last;
};
```

```
struct composite {
    ...

    struct flex middle;

    ... more objects ...
};
```

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member
  - **not the last member – This is deprecated now.**

```
struct composite {
    ...

    struct flex last;
};
```

```
struct composite {
    ...

    struct flex middle;

    ... more objects ...
};
```

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member
  - **not the last member – This is deprecated now.**

```
struct composite {
   ...

   struct flex last;
};
```

```
struct composite {
   ...

   struct flex middle;

   ... more objects ...
};
```

What’s wrong with FAMs in the middle?

– “Compilers do not handle such a case consistently. **Any code relying on this case should be modified to ensure that flexible array members only end up at the ends of structures.**” -GCC Docs.

```
struct composite {
    ...

    struct flex middle;

    ... more objects ...
};
```

So we now have more than **60K**
-Wfamnae warnings to address

So, how did we even get here? - A brief history

- Flexible-Array Transformations - [1] & [0] to C99 [ ]
  - It took us 5 years (2019 – 2024)

So, how did we even get here? - A brief history

- Flexible-Array Transformations - [1] & [0] to C99 [ ]
  - It took us 5 years (2019 – 2024)

```
struct fake_flex_1 {
	...
	size_t count;
	struct foo fake_flex[1];
};
```

```
struct fake_flex_0 {
	...
	size_t count;
	struct foo fake_flex[0];
};
```

So, how did we even get here? - A brief history

- Flexible-Array Transformations - [1] & [0] to C99 [ ]
  - It took us 5 years (2019 – 2024)

```
struct flex {
    ...
    size_t  count;
    struct foo fam[];
};
```

So, how did we even get here? - A brief history

Undefined Behavior – The bug

- e48f129c2f20 ("[SCSI] cxgb3i: convert cdev->l2opt to use…")
- Compilers cannot detect dangerous code like this.

```
struct l2t_data {
        unsigned int nentries;
        struct l2t_entry *rover;
        atomic_t nfree;
        rwlock_t lock;
        struct l2t_entry l2tab[0];
+       struct rcu_head rcu_head;
};
```

So, how did we even get here? - A brief history

Undefined Behavior – The fix

- 76497732932f ("cxgb3/l2t: Fix undefined behavior")
- **Kick-off** of flexible array transformations in the KSPP.
- Bug introduced in **2011**. Fixed in **2019**.

```
struct l2t_data {
        unsigned int nentries;
        struct l2t_entry *rover;
        atomic_t nfree;
        rwlock_t lock;
-       struct l2t_entry l2tab[0];
        struct rcu_head rcu_head;
+       struct l2t_entry l2tab[];
};
```

So, how did we even get here? - A brief history

- Flexible-Array Transformations - [1] & [0] to C99 [ ]
  - It took us 5 years (2019 – 2024)
- [1], [0], [ ] & [N] trailing arrays & fortified memcpy()
  - Fixed __builtin_object_size()
  - Fixed __builtin_dynamic_object_size()

So, how did we even get here? - A brief history

- Flexible-Array Transformations - [1] & [0] to C99 [ ]
  - It took us 5 years (2019 – 2024)
- [1], [0], [ ] & [N] trailing arrays & fortified memcpy()
  - Fixed __builtin_object_size()
  - Fixed __builtin_dynamic_object_size()
  - -fstrict-flex-arrays[=n] – Clang 16 & GCC 13
  - **-fstrict-flex-arrays=3 enabled in Linux 6.5**
    - Only C99 FAMs are considered flex arrays or VLOs.

So, how did we even get here? - A brief history

- The *counted_by* attribute – Clang 18 & GCC 15

So, how did we even get here? - A brief history

- The *counted_by* attribute – Clang 18 & GCC 15
  - Use __builtin_dynamic_object_size() in fortified memcpy().
  - *counted_by* annotations in progress.

So, how did we even get here? - A brief history

- The *counted_by* attribute – Clang 18 & GCC 15
  - Use __builtin_dynamic_object_size() in fortified memcpy().
  - *counted_by* annotations in progress.
  - Ideally, every FAM should be annotated.

Fixing thousands of **-Wflex-array-member-not-at-end** warnings

Fixing thousands of -Wfamnae warnings in Linux

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ... more objects ...
};
```

Fixing thousands of -Wfamnae warnings in Linux

- **Before** first flex-array transformation: 8,000 warnings

```
struct flex {
	...
	size_t  count;
	struct foo fam[] __counted_by(count);
};

struct composite {
	...
	struct flex middle; /* -Wfamnae warning */
	... more objects ...
};
```

Fixing thousands of -Wfamnae warnings in Linux

- **Before** first flex-array transformation: 8,000 warnings
- **After** years of kernel hardening: more than 60,000 warnings

```
struct flex {
        ...
        size_t  count;
        struct foo fam[] __counted_by(count);
};

struct composite {
        ...
        struct flex middle; /* -Wfamnae warning */
        ... more objects ...
};
```

Fixing thousands of -Wfamnae warnings in Linux

- **Before** first flex-array transformation: 8,000 warnings
- **After** years of kernel hardening: more than 60,000 warnings
- 650 unique issues

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ... more objects ...
};
```

Fixing thousands of -Wfamnae warnings in Linux

- **Before** first flex-array transformation: 8,000 warnings
- **After** years of kernel hardening: more than 60,000 warnings
- 650 unique issues – then some patterns started to emerge

```
struct flex {
        ...
        size_t  count;
        struct foo fam[] __counted_by(count);
};

struct composite {
        ...
        struct flex middle; /* -Wfamnae warning */
        ... more objects ...
};
```

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all.**

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all**.

```
 struct wl1251_cmd_header {
     u16 id;
     u16 status;
     /* payload */
     u8 data[];
  } __packed;

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all**.

```
struct wl1251_cmd_header {
        u16 id;
        u16 status;
        /* payload */
        u8 data[];
 } __packed;

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all**.

```
struct wl1251_cmd_header {
    u16 id;
    u16 status;
    /* payload */
    u8 data[];
 } __packed;

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all**.

```
struct wl1251_cmd_header {
    u16 id;
    u16 status;
    /* payload */
    u8 data[];
 } __packed;

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all**.

```
 struct wl1251_cmd_header {
     u16 id;
     u16 status;
     /* payload */
     u8 data[];
  } __packed;

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all**.

```
struct wl1251_cmd_header {
    u16 id;
    u16 status;
    /* payload */
    u8 data[];
 } __packed;

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

-Wflex-array-member-not-at-end

Case 1: **FAMs not used at all**.

- No heap space was allocated for them anywhere.

```
 struct wl1251_cmd_header {
    u16 id;
    u16 status;
    /* payload */
    u8 data[];
  } __packed;

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

-Wflex-array-member-not-at-end

- Case 1: **FAMs not used at all**.
  - f4b09b29f8b4 (“wifi: ti: Avoid a hundred -Wflex-array...”)

```
struct wl1251_cmd_header {
    u16 id;
    u16 status;
-   /* payload */
-   u8 data[];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

```
struct flex {
    int a;
    int b;
    size_t  count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
...


do_something_with(p->middle.a, p->middle.b);
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

```
struct flex {
    int a;
    int b;
    size_t  count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
...

/* We may access the rest of the members in struct flex */
do_something_with(p->middle.a, p->middle.b);
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

```
struct flex {
    int a;
    int b;
    size_t  count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
...

/* We may access the rest of the members in struct flex */
do_something_with(p->middle.a, p->middle.b);
```

But something like

```
... p->middle.fam ...
```

never happens.

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- What can we do about it?

```
struct flex {
    int a; int b;
    size_t count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr {
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

- Case 2: **FAMs never accessed through the composite struct**.
  - Two separate structs: original struct & header struct

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr {
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Two separate structs: original struct & header struct
- New header struct named after original flex struct.

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr {
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Two separate structs: original struct & header struct
- New header struct named after original flex struct.

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Two separate structs: original struct & header struct
- New header struct named after original flex struct.

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Two separate structs: original struct & header struct

```
struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning :( */
    ...
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle;
    ...
};
```

-Wflex-array-member-not-at-end

- Case 2: **FAMs never accessed through the composite struct**.
  - Two separate structs: original struct & header struct

```
struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning :( */
    ...
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle;
    ...
};
```

-Wflex-array-member-not-at-end

- Case 2: **FAMs never accessed through the composite struct**.
  - Two separate structs: original struct & header struct

```
struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning :( */
    ...
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* Life's beautiful! ^.^ */
    ...
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- What’s the problem with this?

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

- Case 2: **FAMs never accessed through the composite struct**.
  - Duplicate code.

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Duplicate code.

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Duplicate code.
- Maintain two independent but basically identical structs.

```
struct flex { /* original struct */
    int a; int b;
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};
```

-Wflex-array-member-not-at-end

- Case 2: **FAMs never accessed through the composite struct**.
  - Use struct_group_tagged()/__struct_group()

-Wflex-array-member-not-at-end

- Case 2: **FAMs never accessed through the composite struct**.
  - Use struct_group_tagged()/__struct_group()

```
struct flex { /* BEFORE */


    int a; int b;
    size_t  count;

    struct foo fam[] __counted_by(count);
};

struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* FAM is gone! ^.^ */
    ...
} *p;
```

The **struct_group()** family of helpers

Created by Kees Cook and Keith Packard

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                     \
                struct { MEMBERS };                 \
                struct TAG { MEMBERS } NAME;        \
        }
```

The **struct_group()** family of helpers

Created by Kees Cook and Keith Packard

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
                struct { MEMBERS };                \
                struct TAG { MEMBERS } NAME;       \
        }
```

The **struct_group()** family of helpers

- Created by Kees Cook and Keith Packard
  - Access each member directly or via the named struct.

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                     \
                struct { MEMBERS };                 \
                struct TAG { MEMBERS } NAME;        \
        }
```

The **struct_group()** family of helpers

Created by Kees Cook and Keith Packard

- Access each member directly or via the named struct.
- Create a new struct type and define an identifier for the group

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
               struct { MEMBERS };                 \
               struct TAG { MEMBERS } NAME;        \
        }
```

The **struct_group()** family of helpers

Created by Kees Cook and Keith Packard

- Access each member directly or via the named struct.
- Create a new struct type and define an identifier for the group – via which we can gain bounds-checking.

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
                struct { MEMBERS };                \
                struct TAG { MEMBERS } NAME;       \
        }
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* FAM is gone! ^.^ */
    ...
} *p;
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

– Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* FAM is gone! ^.^ */
    ...
} *p;
```

```
{
    p->middle.a
    p->middle.b
    p->middle.count
}
        ==
{
    p->middle.hdr.a
    p->middle.hdr.b
    p->middle.hdr.count
}
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

– 5c4250092fad (“wifi: mwl8k: Avoid -Wflex-array-…”)

```
 struct mwl8k_cmd_pkt {
-	__le16	code;
-	__le16	length;
-	__u8	seq_num;
-	__u8	macid;
-	__le16	result;
+	__struct_group(mwl8k_cmd_pkt_hdr, hdr, __packed,
+		__le16	code;
+		__le16	length;
+		__u8	seq_num;
+		__u8	macid;
+		__le16	result;
+	);
	char payload[];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 2: **FAMs never accessed through the composite struct**.

– 5c4250092fad (“wifi: mwl8k: Avoid -Wflex-array-…”)

```
 struct mwl8k_cmd_pkt {
-	__le16	code;
-	__le16	length;
-	__u8	seq_num;
-	__u8	macid;
-	__le16	result;
+	__struct_group(mwl8k_cmd_pkt_hdr, hdr, __packed,
+		__le16	code;
+		__le16	length;
+		__u8	seq_num;
+		__u8	macid;
+		__le16	result;
+	);
	char payload[];
 } __packed;
```

-Wflex-array-member-not-at-end

- 5c4250092fad (“wifi: mwl8k: Avoid -Wflex-array-…”)
- Replace *mwl8k_cmd_pkt* with *mwl8k_cmd_pkt_hdr*

```
 struct mwl8k_cmd_rf_antenna {
-  struct mwl8k_cmd_pkt header;
+  struct mwl8k_cmd_pkt_hdr header;
   __le16 antenna;
   __le16 mode;
 } __packed;
```

-Wflex-array-member-not-at-end

- 5c4250092fad (“wifi: mwl8k: Avoid -Wflex-array-…”)
- Replace *mwl8k_cmd_pkt* with *mwl8k_cmd_pkt_hdr*

```
  struct mwl8k_cmd_rf_antenna {
-   struct mwl8k_cmd_pkt header;
+   struct mwl8k_cmd_pkt_hdr header;
    __le16 antenna;
    __le16 mode;
  } __packed;
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
struct flex_struct {
    ...
    size_t  count;
    struct foo flex_array[] __counted_by(count);
};

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
struct flex_struct {
    ...
    size_t  count;
    struct foo flex_array[] __counted_by(count);
};

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
struct flex_struct {
    ...
    size_t  count;
    struct foo flex_array[] __counted_by(count);
};

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

- flex_array and fixed_array share the same address in memory - in the best scenario.
- Both form an implicit union.

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-  struct ima_digest_data hdr;
+  struct ima_digest_data_hdr hdr;
   u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-  struct ima_digest_data hdr;
+  struct ima_digest_data_hdr hdr;
   u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- However, **FAM digest** is accessed at run-time.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- However, **FAM digest** is accessed at run-time.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- However, **FAM digest** is accessed at run-time.

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
struct ima_max_digest_data hash;
...
/* read data from the FAM digest */
memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- However, **FAM digest** is accessed at run-time.

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
struct ima_max_digest_data hash;
...
/* read data from the FAM digest */
memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- However, **FAM digest** is accessed at run-time.

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;


  struct ima_max_digest_data hash;
  ...
  /* read data from the FAM digest */
  memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

– However, **FAM digest** is accessed at run-time.

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;


  struct ima_max_digest_data hash;
  ...
  /* read data from the FAM digest */
  memcpy(digest_hash, ~~hash.hdr.digest~~, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.
- Access FAM through that pointer.

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.
- Access FAM through that pointer.

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                         struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

        /* read data from the FAM digest */
-       memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+       memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.
- Access FAM through that pointer.

```
     struct ima_max_digest_data hash; /* struct with implicit union */
+    struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                             struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

         /* read data from the FAM digest */
-        memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+        memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.
- Access FAM through that pointer.

```
     struct ima_max_digest_data hash; /* struct with implicit union */
+    struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                             struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

         /* read data from the FAM digest */
-        memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+        memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.
- Access FAM through that pointer.

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                            struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

        /* read data from the FAM digest */
-       memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+       memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                            struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                                struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+  /* New members must be added within the __struct_group() macro below. */
+  __struct_group(ima_digest_data_hdr, hdr, __packed,
      u8 algo;
      u8 length;
      ...
+  );
   u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-  struct ima_digest_data hdr;
+  struct ima_digest_data_hdr hdr;
   u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
  struct ima_max_digest_data hash; /* struct with implicit union */
+ struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                          struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                               struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                                struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                                  struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                            struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
        u8 algo;
        u8 length;
        ...
+   );
    u8 digest[];
 } __packed;
```

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                                      struct ima_digest_data, hdr);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.
- Access FAM through that pointer.
- 38aa3f5ac6d2 (“integrity: Avoid -Wflex-array-member...”)

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                                 struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

        /* read data from the FAM digest */
-       memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+       memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: **Implicit unions** between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.
- Access FAM through that pointer.
- 38aa3f5ac6d2 (“integrity: Avoid -Wflex-array-member...”)

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                            struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

        /* read data from the FAM digest */
-       memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+       memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct flex_struct {
    ...
    size_t  count;
    struct foo flex_array[];
};


int some_function(...)
{
    struct {
        struct flex_struct flex; /* on-stack -Wfamnae warning */
        struct foo fixed_array[10];
    } obj = ...
    ...
}
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct flex_struct {
    ...
    size_t  count;
    struct foo flex_array[];
};


int some_function(...)
{
    struct {
        struct flex_struct flex; /* on-stack -Wfamnae warning */
        struct foo fixed_array[10];
    } obj = ...
    ...
}
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct flex_struct {
	...
	size_t  count;
	struct foo flex_array[]; /* flex-array member */
};


int some_function(...)
{
	struct {
		struct flex_struct flex; /* on-stack -Wfamnae warning */
		struct foo fixed_array[10]; /* fixed-size array */
	} obj = ...
	...
}
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct fun_admin_bind_req {
    struct fun_admin_req_common common;
    struct fun_admin_bind_entry entry[];
};

int fun_bind(...)
{
    struct {
        struct fun_admin_bind_req req;/* on-stack -Wfamnae warning */
        struct fun_admin_bind_entry entry[2];
    } cmd = ...
    ...
}
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct fun_admin_bind_req {
    struct fun_admin_req_common common;
    struct fun_admin_bind_entry entry[];
};

int fun_bind(...)
{
    struct {
        struct fun_admin_bind_req req;/* on-stack -Wfamnae warning */
        struct fun_admin_bind_entry entry[2];
    } cmd = ...
    ...
}
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct fun_admin_bind_req {
    struct fun_admin_req_common common;
    struct fun_admin_bind_entry entry[];     /* flex-array member */
};

int fun_bind(...)
{
    struct {
        struct fun_admin_bind_req req;/* on-stack -Wfamnae warning */
        struct fun_admin_bind_entry entry[2]; /* fixed-size array */
    } cmd = ...
    ...
}
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-	struct {
-		struct fun_admin_bind_req req;
-		struct fun_admin_bind_entry entry[2];
-	} cmd = {
-		.req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-						sizeof(cmd)),
-		.entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-		.entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-	};
+	DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+	cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+					__struct_size(cmd));
+	cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+	cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-	struct {
-		struct fun_admin_bind_req req;
-		struct fun_admin_bind_entry entry[2];
-	} cmd = {
-		.req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-							 sizeof(cmd)),
-		.entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-		.entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-	};
+	DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+	cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+						 __struct_size(cmd));
+	cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+	cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                      sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                  __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-	struct {
-		struct fun_admin_bind_req req;
-		struct fun_admin_bind_entry entry[2];
-	} cmd = {
-		.req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-							 sizeof(cmd)),
-		.entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-		.entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-	};
+	DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+	cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+						 __struct_size(cmd));
+	cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+	cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                        __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                        __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                  sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                              __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                           sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                       __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

-Wflex-array-member-not-at-end

Case 4: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

- We use **DECLARE_FLEX()** and **DECLARE_RAW_FLEX()** helpers.
- Some examples:
  - 6c85a13b133f (“platform/chrome: cros_ec_proto:…”)
  - 4d69c58ef2e4 (“fsnotify: Avoid -Wflex-array-mem…”)
  - 215c4704208b (“Bluetooth: L2CAP: Avoid -Wflex-...”)

Other issues: struct_group(), UAPI headers & C++

Other issues: struct_group(), UAPI headers & C++

```
--- a/include/uapi/linux/ethtool.h
+++ b/include/uapi/linux/ethtool.h

struct ethtool_link_settings {
-	__u32	cmd;
-	__u32	speed;
	...
+	__struct_group(ethtool_link_settings_hdr, hdr, /* no attrs */,
+		__u32	cmd;
+		__u32	speed;
		 ...
+	);
	__u32	link_mode_masks[];
 };
```

Other issues: struct_group(), UAPI headers & C++

“Possibly a very noob question, but I'm updating **a C++ library with new headers** and I think **this makes it no longer compile.**” - Jakub Kicinski (NetDev maintainer)

```
--- a/include/uapi/linux/ethtool.h
+++ b/include/uapi/linux/ethtool.h

struct ethtool_link_settings {
-    __u32    cmd;
-    __u32    speed;
     ...
+    __struct_group(ethtool_link_settings_hdr, hdr, /* no attrs */,
+        __u32    cmd;
+        __u32    speed;
          ...
+    );
     __u32    link_mode_masks[];
 };
```

Other issues: struct_group(), UAPI headers & C++

```
$ g++ /tmp/t.cpp -I../linux -o /dev/null -c -W -Wall -O2
... ‘struct ethtool_link_settings::<unnamed union>::ethtool_link_settings_hdr’
invalid; an anonymous union may only have public non-static data members [-
fpermissive]
 2515 |         __struct_group(ethtool_link_settings_hdr, hdr, /* no attrs */,
      |                        ^~~~~~~~~~~~~~~~~~~~~~~~~
```

```
struct ethtool_link_settings {
-   __u32   cmd;
-   __u32   speed;
    ...
+   __struct_group(ethtool_link_settings_hdr, hdr, /* no attrs */,
+       __u32   cmd;
+       __u32   speed;
         ...
+   );
    __u32   link_mode_masks[];
 };
```

Other issues: struct_group(), UAPI headers & C++

```
$ g++ /tmp/t.cpp -I../linux -o /dev/null -c -W -Wall -O2
... ‘struct ethtool_link_settings::<unnamed union>::ethtool_link_settings_hdr’
invalid; an anonymous union may only have public non-static data members [-
fpermissive]
 2515 |         __struct_group(ethtool_link_settings_hdr, hdr, /* no attrs */,
      |                        ^~~~~~~~~~~~~~~~~~~~~~~~~
```

```
 struct ethtool_link_settings {
-    __u32   cmd;
-    __u32   speed;
     ...
+    __struct_group(ethtool_link_settings_hdr, hdr, /* no attrs */,
+        __u32   cmd;
+        __u32   speed;
         ...
+    );
     __u32   link_mode_masks[];
 };
```

Other issues: struct_group(), UAPI headers & C++

C++ & type declarations inside an anonymous union

```
#define __struct_group(TAG, NAME, ATTRS, MEMBERS...)\
        union {                                     \
                struct { MEMBERS } ATTRS;           \
                struct TAG { MEMBERS } ATTRS NAME;  \
        }
```

Other issues: struct_group(), UAPI headers & C++

C++ & type declarations inside an anonymous union

```
#define __struct_group(TAG, NAME, ATTRS, MEMBERS...)\
         union {                                    \
                 struct { MEMBERS } ATTRS;          \
                 struct TAG { MEMBERS } ATTRS NAME; \
        }
```

Other issues: struct_group(), UAPI headers & C++

C++ & type declarations inside an anonymous union

```
#define __struct_group(TAG, NAME, ATTRS, MEMBERS...)\
         union {                                    \
                struct { MEMBERS } ATTRS;           \
                struct TAG { MEMBERS } ATTRS NAME;  \
        }
```

Other issues: struct_group(), UAPI headers & C++

C++ & type declarations inside an anonymous union

- I wanted to revert those changes.
- Kees proposed a case-by-case fix, and sent a patch.

Other issues: struct_group(), UAPI headers & C++

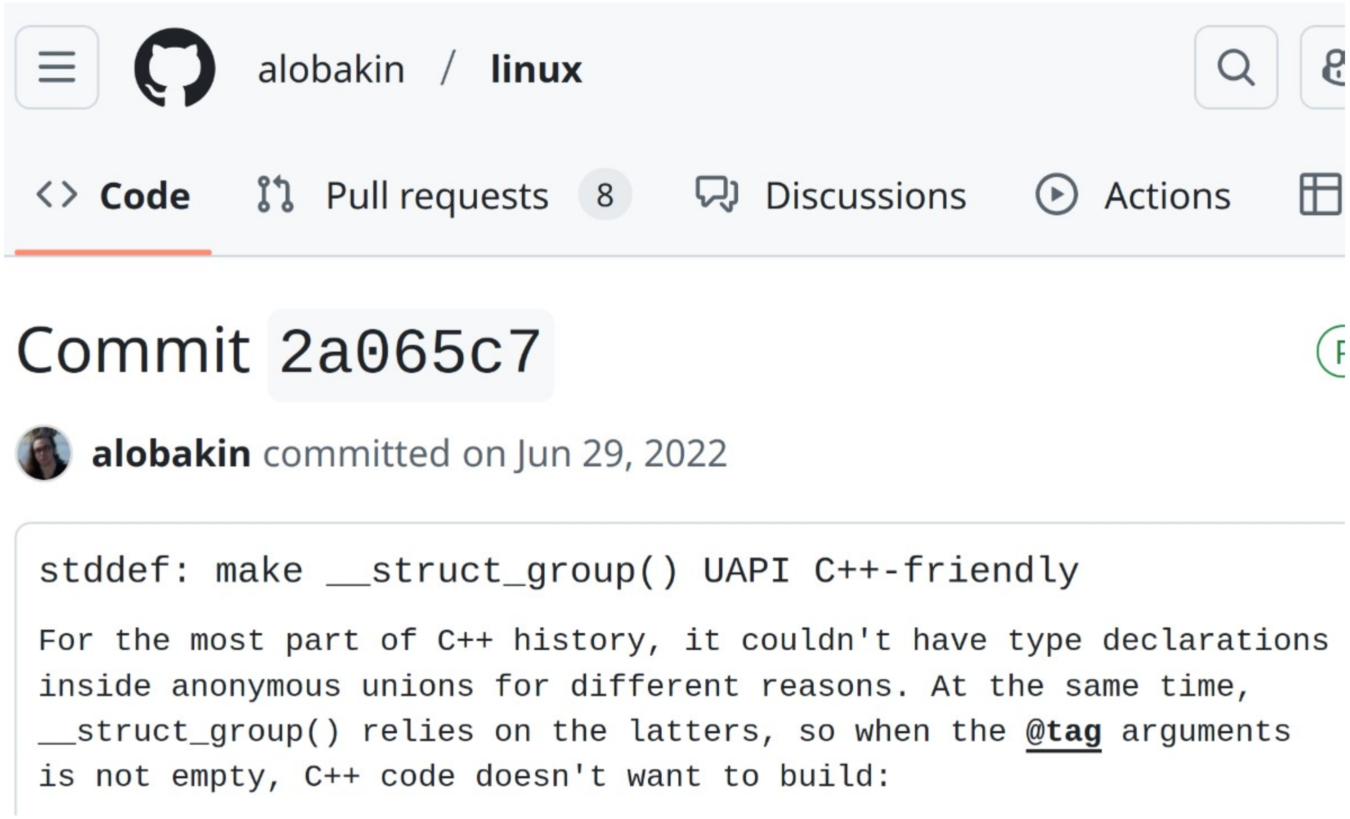
C++ & type declarations inside an anonymous union

“**I worked around that like this in the past**: [0]
As I'm not sure it would be fine to fix every such
occurrence manually by open-coding.
What do you think?

[0] https://github.com/alobakin/linux/commit/2a065c7bae821f5fa85fff6f97fbbd460f4aa0f3”
– Alexander Lobakin

Other issues: struct_group(), UAPI headers & C++

C++ & type declarations inside an anonymous union


alobakin / linux

Code Pull requests 8 Discussions Actions

Commit 2a065c7

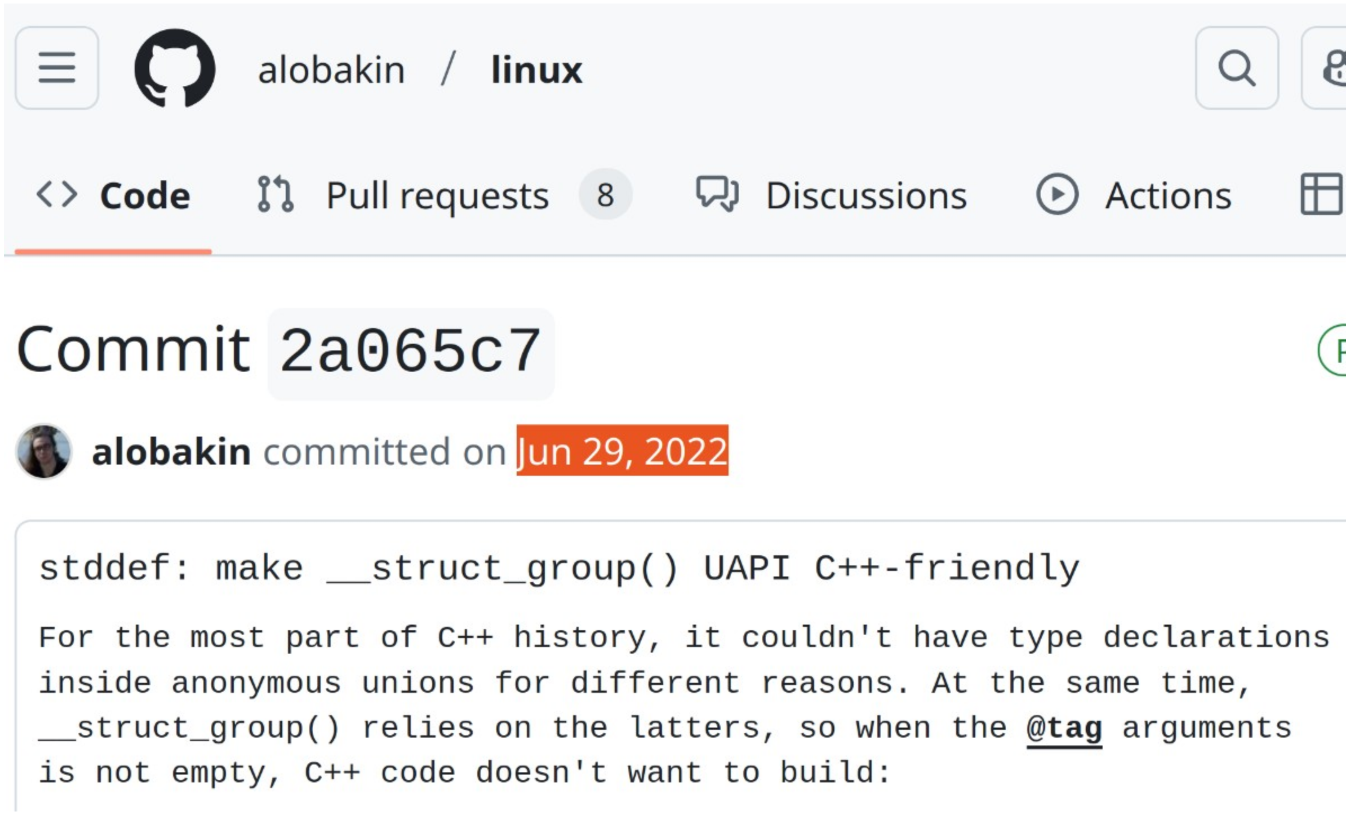
alobakin committed on Jun 29, 2022

```
stddef: make __struct_group() UAPI C++-friendly

For the most part of C++ history, it couldn't have type declarations
inside anonymous unions for different reasons. At the same time,
__struct_group() relies on the latters, so when the @tag arguments
is not empty, C++ code doesn't want to build:
```

Other issues: struct_group(), UAPI headers & C++

C++ & type declarations inside an anonymous union



Commit 2a065c7

alobakin committed on Jun 29, 2022

```
stddef: make __struct_group() UAPI C++-friendly

For the most part of C++ history, it couldn't have type declarations
inside anonymous unions for different reasons. At the same time,
__struct_group() relies on the latters, so when the @tag arguments
is not empty, C++ code doesn't want to build:
```

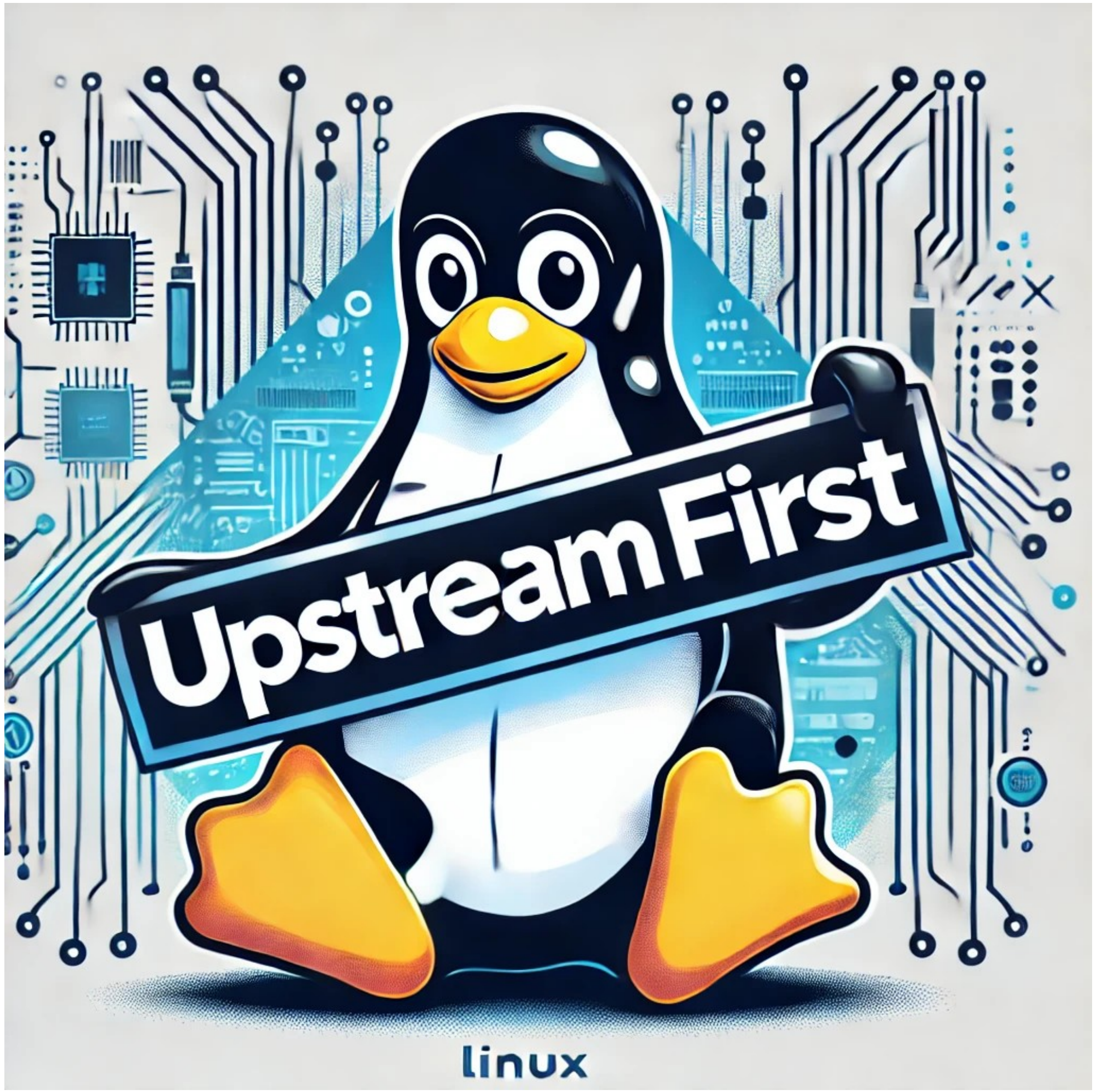
Other issues: struct_group(), UAPI headers & C++

C++ & type declarations inside an anonymous union

“**I worked around that like this in the past:** [0]
As I'm not sure it would be fine to fix every such
occurrence manually by open-coding.
What do you think?

[0] https://github.com/alobakin/linux/commit/2a065c7bae821f5fa85fff6f97fbbd460f4aa0f3”
– Alexander Lobakin

commit 724c6ce38bba (“stddef: make __struct_group() UAPI C++-friendly”)


Upstream First
linux

What’s next in the KSPP?

What’s next in the KSPP?

Extend *__counted_by* to pointers in structures :)

What’s next in the KSPP?

Extend __*counted_by* to pointers in structures :)

```
struct bounded_p {
    ...
    size_t count;
    struct foo *p __counted_by(.count);
};
```

What’s next in the KSPP?

Extend __*counted_by* to pointers in structures :)

```
struct bounded_p {
   ...
   size_t count;
   struct foo *p __counted_by(.count);
};
```

What’s next in the KSPP?

Extend __*counted_by* to pointers in structures :)

```
struct bounded_p {
    ...
    struct foo *p __counted_by(.count);
    size_t count;
};
```

What’s next in the KSPP?

Extend __*counted_by* to pointers in structures :)

- Coming in GCC 16 (**Qing Zhao**) & Clang 19 (**Bill Wendling**)

```
struct bounded_p {
    ...
    struct foo *p __counted_by(.count);
    size_t count;
};
```

What’s next in the KSPP?

Extend __*counted_by* to pointers in structures :)

- Coming in GCC 16 (**Qing Zhao**) & Clang 19 (**Bill Wendling**)

```
struct some_header {
    ...
    size_t count;
    ...
}

struct bounded_flex {
    ...
    struct some_header hdr;
    struct foo fam[] __counted_by(.hdr.count);
};
```

Conclusions

Conclusions

A simple three-step solution for the complex case:

Conclusions

A simple three-step solution for the complex case:

- Use **struct_group_tagged()** to create a new tagged struct.
  - This groups together all members in the flex struct **except the FAM**.

Conclusions

A simple three-step solution for the complex case:

- Use **struct_group_tagged()** to create a new tagged struct.
  - This groups together all members in the flex struct **except the FAM**.
- **Change the type** of the conflicting object to the newly created tagged struct.

Conclusions

A simple three-step solution for the complex case:

- Use **struct_group_tagged()** to create a new tagged struct.
    - This groups together all members in the flex struct **except the FAM**.
- **Change the type** of the conflicting object to the newly created tagged struct.
- Use **container_of()** to retrieve a pointer to the flex struct when needed.
    - Access the **FAM** via this pointer if necessary.

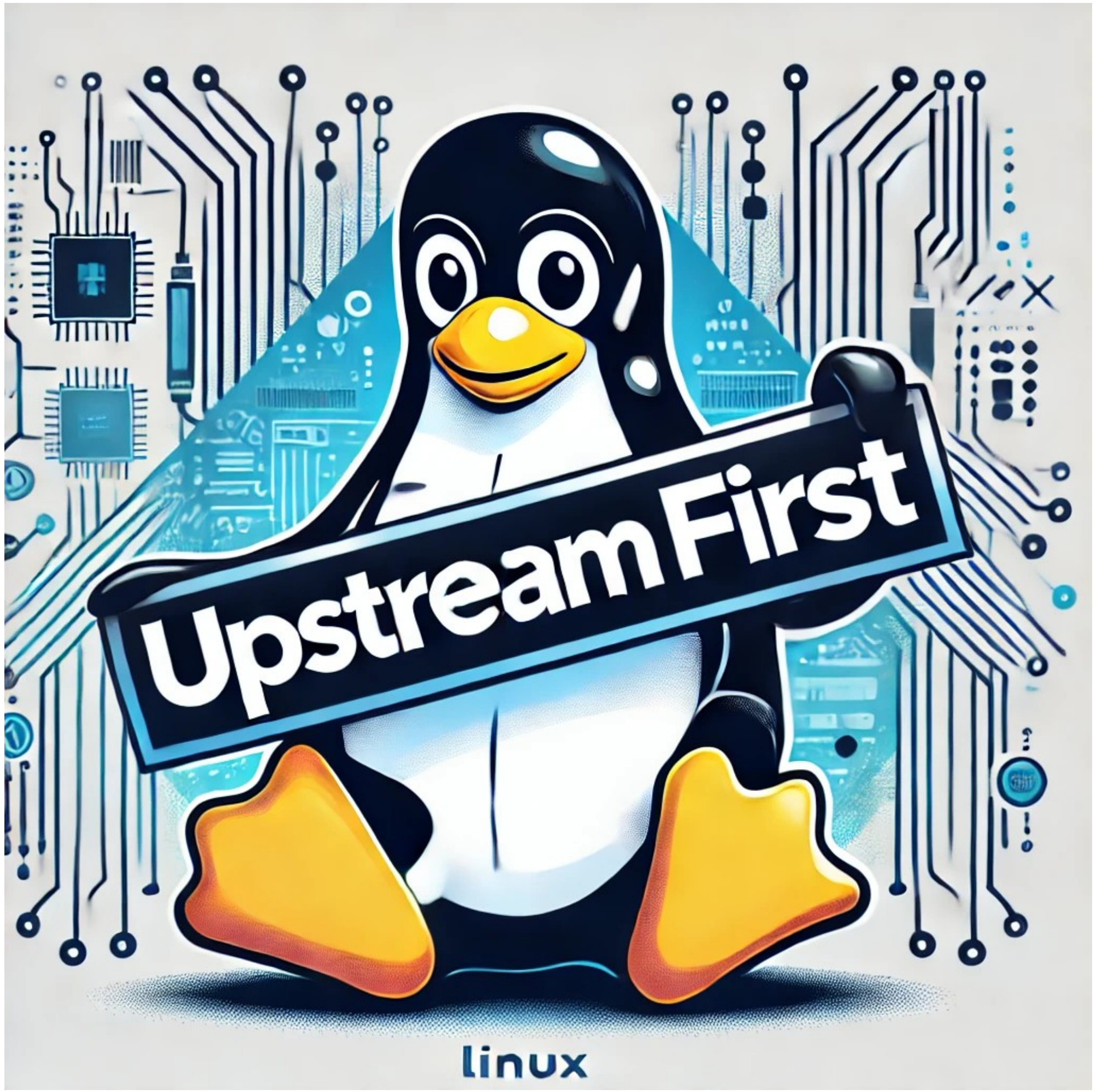
Conclusions

For implicit unions on the stack:

- Use **DECLARE_FLEX()** when the FAM is annotated with __counted_by().
- We can use **DECLARE_RAW_FLEX()** in any other case.

Conclusions

- Clear strategy to enable **-Wflex-array-member-not-at-end** in mainline, soon.
- Dozens of patches are already in mainline.
- Down to ~300 (from ~650) unique warnings.
- ~30% of total warnings addressed so far.


Upstream First
linux

Thank you, Adelaide!

Gustavo A. R. Silva
gustavoars@kernel.org
fosstodon.org/@gustavoars

By @shidokou

Some resources

- How to use the new counted_by attribute in C and Linux (2024)
- Enhancing spatial safety: Better array-bounds checking in C (and Linux) (2024)
- Gaining bounds-checking on trailing arrays in the Linux Kernel (2024)
- Flexible-Array Transformations and Array-bounds checking (2022)