$\S 1$  C CWEB OUTPUT 1

```
list_entry - get the struct for this entry.
              the &struct list_head pointer.
  ptr:
              the type of the struct this is embedded in.
  type:
  member: the name of the list_head within the struct.
#define list_entry(ptr, type, member)
         container\_of(ptr, type, member)
2. list_prev_entry - get the prev element in list.
              the type * to cursor.
  pos:
             the name of the list_head within the struct.
  member:
#define list_prev_entry(pos, member)
         list\_entry((pos) \neg member.prev, typeof(*(pos)), member)
3. list_next_entry - get the next element in list.
              the type * to cursor.
             the name of the list_head within the struct.
  member:
#define list_next_entry(pos, member)
         list\_entry((pos) \rightarrow member.next, typeof(*(pos)), member)
4. list_last_entry - get the last element from a list.
              the list head to take the element from.
  ptr:
              the type of the struct this is embedded in.
  type:
  member:
             the name of the list_head within the struct.
\#define list\_last\_entry(ptr, type, member)
         list\_entry((ptr) \rightarrow prev, type, member)
5. list_first_entry - get the first element from a list.
  ptr:
              the list head to take the element from.
              the type of the struct this is embedded in.
  type:
  member: the name of the list_head within the struct.
\#define list\_first\_entry(ptr, type, member)
         list\_entry((ptr) \neg next, type, member)
6. list_entry_is _head - test if the entry points to the head of the list.
              the type * to cursor.
  pos:
  head:
              the head for your list.
  member: the name of the list_head within the struct.
#define list_entry_is_head(pos, head, member)
         (\&pos \neg member \equiv (head))
7. list_for_each_entry - iterate over list of given type.
              the type * to cursor.
  pos:
  head:
              the head for your list.
  member: the name of the list_head within the struct.
#define list_for_each_entry(pos, head, member)
         for (pos = list\_first\_entry(head, typeof(*pos), member);
         \neg list\_entry\_is\_head(pos, head, member);
         pos = list\_next\_entry(pos, member))
```

2 CWEB OUTPUT C §8

list\_for\_each\_entry\_reverse - iterate backwards over list of given type. the type \* to cursor. pos: head: the head for your list. member: the name of the list\_head within the struct. #define list\_for\_each\_entry\_reverse(pos, head, member) for  $(pos = list\_last\_entry(head, typeof(*pos), member);$  $\neg list\_entry\_is\_head(pos, head, member);$  $pos = list\_prev\_entry(pos, member))$ 9. list\_for\_each\_entry\_safe - iterate over list of given type safe against removal of list entry. the type \* to use as a loop cursor. pos: another type \* to use as temporary storage. n: the head for your list. head: member: the name of the list\_head within the struct. #**define**  $list\_for\_each\_entry\_safe(pos, n, head, member)$ for  $(pos = list\_first\_entry(head, typeof(*pos), member),$  $n = list\_next\_entry(pos, member);$  $\neg list\_entry\_is\_head(pos, head, member);$  $pos = n, n = list\_next\_entry(n, member))$ 

 $\S10$  C index 3

## 10. Index.

```
container_of: 1.
head: 6, 7, 8, 9.
list_entry: 1, 2, 3, 4, 5.
list_entry_is_head: 6, 7, 8, 9.
list_first_entry: 5, 7, 9.
list_for_each_entry: 7.
list_for_each_entry_reverse: 8.
list_for_each_entry_safe: 9.
list_last_entry: 4, 8.
list_next_entry: 2, 8.
member: 1, 2, 3, 4, 5, 6, 7, 8, 9.
next: 3, 5.
pos: 2, 3, 6, 7, 8, 9.
prev: 2, 4.
ptr: 1, 4, 5.
typeof: 2, 3, 7, 8, 9.
```

	Section	Page
Index	10	3