struct list_head {
    struct list_head *next, *prev;
};

static inline void INIT_LIST_HEAD(struct list_head *list) {
    list->next = list;
    list->prev = list;
}

/**
 * list_entry - get the struct for this entry
 * @ptr: the &struct list_head pointer.
 * @type: the type of the struct this is embedded in.
 * @member: the name of the list_struct within the struct.
 */
#define list_entry(ptr, type, member) \    
    const typeof((type *)0)->member *)__mptr = (ptr);
    (type *){(char *)__mptr - offsetof(type, member)};

#define container_of(ptr, type, member) ({
    const typeof((type *)0)->member *)__mptr = (ptr);
    (type *)(char *)__mptr - offsetof(type, member));

#define offsetof(TYPE, MEMBER) ((size_t) &((TYPE *)0)->MEMBER)

/*Typical usage of list_heads and the list_entry macro*/
struct list_head *report_list = /*gotten from somewhere*/
struct hid_report *report = list_entry(report_list->next, struct hid_report, list);

struct hid_report {
    unsigned id;
    unsigned type;
    struct hid_field *field[HID_MAX_FIELDS];
    struct list_head list;
    unsigned maxfield;
    unsigned size;
    struct hid_device *device;
};

What this macro produces. /* id of this report */ /* report type */ /* fields of the report */ /* maximum valid field index */ /* size of the report (bits) */ /* associated device */